OpenVOS System Administration: Administering and Customizing a System

Stratus Technologies
R281-16

Notice

Manual Name: *OpenVOS System Administration: Administering and Customizing a System* (R281)

Part Number: R281
Revision Number: 16
OpenVOS Release Number: 19.3.1
Publication Date: January 2022

Stratus Technologies, Inc.
5 Mill and Main Place, Suite 500
Maynard, Massachusetts 01754-2660

Contents

Figures

Tables

Preface

The manual *OpenVOS System Administration: Administering and Customizing a System* (R281) documents the procedures for administering Stratus modules.

This manual is intended for system administrators.

Manual Version

This manual is a revision. Change bars, which appear in the margin, note the specific changes to text since the previous publication of this manual.

This revision includes support for OpenVOS Release 19.3.1, including support for ftServer V 2810, V 4820, and V 6832 systems, as well as a correction to the `maint_request` documentation.

NOTES

1. Contact your account representative for information about the availability of ftServer V 6728 systems on OpenVOS Release 19.2.*x* and later.
2. Contact your account representative for information about the availability of OpenVOS 19.*x* support for the network I/O enclosure.

Related Manuals

See the following Stratus manuals for related documentation.

- *OpenVOS PL/I Subroutines Manual* (R005)
- *OpenVOS C Subroutines Manual* (R068)
- *OpenVOS System Analysis Manual* (R073)
- *OpenVOS Commands User's Guide* (R089)
- *OpenVOS Commands Reference Manual* (R098)
- *VOS Transaction Processing Facility Guide* (R215)
- *OpenVOS System Administration: Starting Up and Shutting Down a Module or System* (R282)
- *OpenVOS System Administration: Registration and Security* (R283)

- *OpenVOS System Administration: Disk and Tape Administration* (R284)
- *OpenVOS System Administration: Backing Up and Restoring Data* (R285)
- *VOS System Administration: Administering the Spooler Facility* (R286)
- *OpenVOS System Administration: Configuring a System* (R287)
- *VOS Administrator's Guide for Open StrataLINK* (R388)
- *OpenVOS STREAMS TCP/IP Administrator's Guide* (R419)
- *Product Release Bulletin: VOS Configuration for the V105 Terminal* (R439)
- *Using OpenVOS Extended Names* (R631)
- *Stratus ftServer V Series Systems: V 6728 Hardware Supplement* (R790)
- site planning guides:

  Stratus ftServer V 2404, V 4408, and V 6408 Systems: Site Planning Guide (R645)

  Stratus ftServer V 6512 System: Site Planning Guide (R675)

  Stratus ftServer V 6624 System: Site Planning Guide (R763)

  Stratus ftServer V 2608, V 4612, and V 6616 Systems: Site Planning Guide (R772)

  Stratus ftServer V 2810, V 4820, and V 6832 Systems: Site Planning Guide (R794)

- operation and maintenance guides:

  Stratus ftServer V 2404, V 4408, and V 6408 Systems: Operation and Maintenance Guide (R646)

  Stratus ftServer V 6512 System: Operation and Maintenance Guide (R674)

  Stratus ftServer V 6624 System: Operation and Maintenance (R764)

  Stratus ftServer V 2608, V 4612, and V 6616 Systems: Operation and Maintenance (R773)

  Stratus ftServer V 2810, V 4820, and V 6832 Systems: Operation and Maintenance (R795)

Notation Conventions

This manual uses the following notation conventions.

Warnings, Cautions, Notices, and Notes

Warnings, cautions, notices, and notes provide special information and have the following meanings:

WARNING

A warning indicates a hazardous situation that, if not avoided, could result in death or serious injury.

AVERTISSEMENT

Un avertissement indique une situation dangereuse qui, si pas évitée, pourrait entraîner la mort ou des blessures graves.

CAUTION

A caution indicates a hazardous situation that, if not avoided, could result in minor or moderate injury.

MISE EN GARDE

Une mise en garde indique une situation dangereuse qui, si pas évitée, pourrait entraîner des blessures mineures ou modérées.

NOTICE

A notice indicates information that, if not acted on, could result in damage to a system, hardware device, program, or data, but does not present a health or safety hazard.

NOTE

A note provides important information about the operation of an ftServer system or related equipment or software.

Typographical Conventions

The following typographical conventions are used in this manual:

- Italics introduces or defines new terms. For example:

  The *master disk* is the name of the member disk from which the module was booted.

- Boldface emphasizes words in text. For example:

  Every module **must** have a copy of the `module_start_up.cm` file.

- Monospace represents text that would appear on your terminal's screen (such as commands, subroutines, code fragments, and names of files and directories). For example:

  ```
  change_current_dir (master_disk)>system>doc
  ```

- Monospace italic represents terms that are to be replaced by literal values. In the following example, the user must replace the monospace-italic term with a literal value.

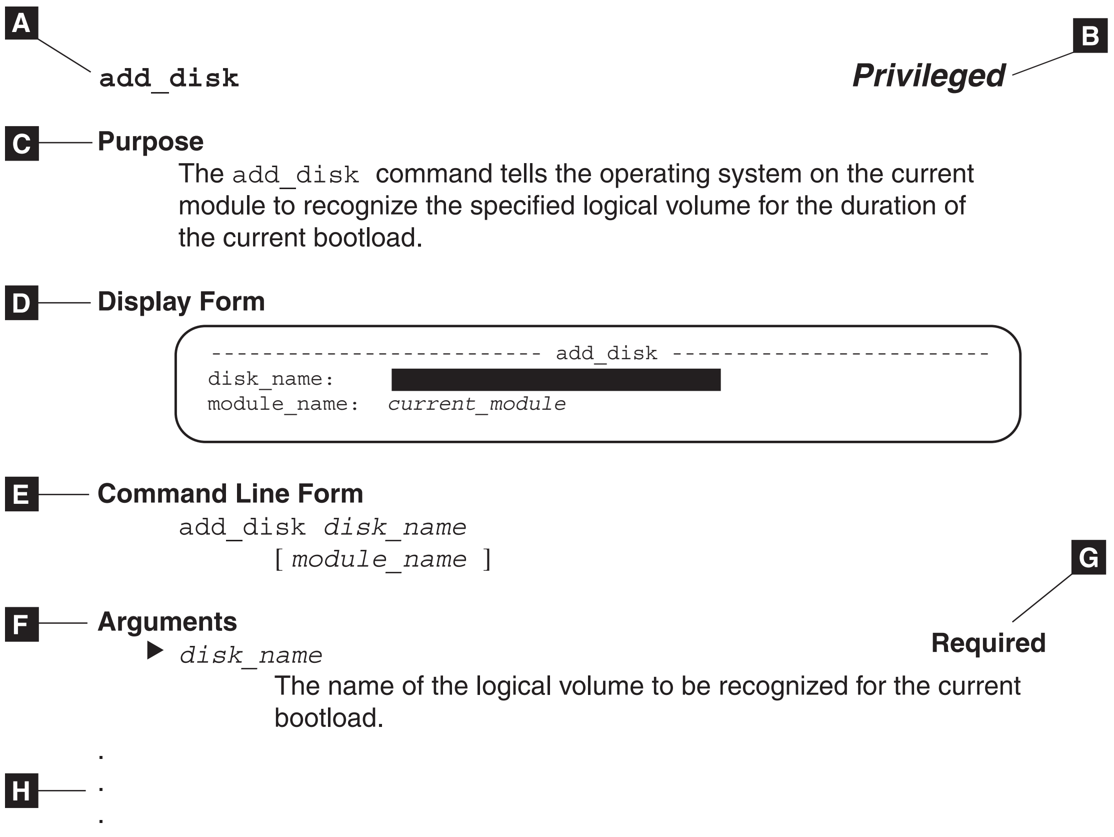
  ```
  list_users -module module_name
  ```

- Monospace bold represents user input in examples and figures that contain both user input and system output (which appears in monospace). For example:

  ```
  display_access_list system_default

  %dev#m1>system>acl>system_default
  ```

Format for Commands and Requests

Stratus manuals use the following format conventions for documenting commands and requests. (A *request* is typically a command used within a subsystem, such as `analyze_system`.) Note that the command and request descriptions do not necessarily include each of the following sections.



`add_disk` ***Privileged***

Purpose

The `add_disk` command tells the operating system on the current module to recognize the specified logical volume for the duration of the current bootload.

Display Form

```
-------------------------- add_disk -------------------------
disk_name:
module_name:   current_module
```

Command Line Form

```
add_disk disk_name
      [ module_name ]
```

Arguments

▶ *`disk_name`* **Required**

The name of the logical volume to be recognized for the current bootload.

.
.
.

A `name`

The name of the command or request is at the top of the first page of the description.

B *Privileged*

This notation appears after the name of a command or request that can be issued only from a privileged process.

C Purpose

Explains briefly what the command or request does.

D Display Form

Shows the form that is displayed when you type the command or request name followed by `-form` or when you press the key that performs the `DISPLAY FORM` function. Each field in the form represents a command or request argument. If an argument has a default value, that value is displayed in the form.

The following table explains the notation used in display forms.

The Notation Used in Display Forms

Notation	Meaning
███████	Required field with no default value.
█	The cursor, which indicates the current position on the screen. For example, the cursor may be positioned on the first character of a value, as in `all`.
`current_user` *`current_module`* *`current_system`* *`current_disk`*	The default value is the current user, module, system, or disk. The actual name is displayed in the display form of the command or request.

E Command-Line Form

Shows the syntax of the command or request with its arguments. You can display an online version of the command-line form of a command or request by typing the command or request name followed by `-usage`.

The following table explains the notation used in command-line forms. In the table, the term *multiple values* refers to explicitly stated separate values, such as two or more object names. Specifying multiple values is **not** the same as specifying a star name. When you specify multiple values, you must separate each value with a space.

The Notation Used in Command-Line Forms

Notation	Meaning
`argument_1`	Required argument.
`argument_1``...`	Required argument for which you can specify multiple values.
$\left\{\begin{matrix}\mathit{argument\_1} \\ \mathit{argument\_2}\end{matrix}\right\}$	Set of arguments that are mutually exclusive; you must specify one of these arguments.
$\left[\mathit{argument\_1}\right]$	Optional argument.
$\left[\mathit{argument\_1}\right]$`...`	Optional argument for which you can specify multiple values.
$\left[\begin{matrix}\mathit{argument\_1} \\ \mathit{argument\_2}\end{matrix}\right]$	Set of optional arguments that are mutually exclusive; you can specify only one of these arguments.

Notation	**Meaning**
Note: Dots, brackets, and braces are not literal characters; you should **not** type them. Any list or set of arguments can contain more than two elements. Brackets and braces are sometimes nested.	

F Arguments

Describes the command or request arguments. The following table explains the notation used in argument descriptions.

G The Notation Used in Argument Descriptions

Notation	**Meaning**
CYCLE	This argument has predefined values. In the display form, you display these values in sequence by pressing the key that performs the `CYCLE` function.
Required	You cannot issue the command or request without specifying a value for this argument. If an argument is required but has a default value, it is not labeled **Required** since you do not need to specify it in the command-line form. However, in the display form, a required field must have a value—either the displayed default value or a value that you specify.
(Privileged)	Only a privileged process can specify a value for this argument.

H The following additional headings may appear in the command or request description: Explanation, Error Messages, Examples, and Related Information.

Explanation

Explains how to use the command or request and provides supplementary information.

Error Messages

Lists common error messages with a short explanation.

Examples

Illustrates uses of the command or request.

Related Information

Refers you to related information (in this manual or other manuals), including descriptions of commands, subroutines, and requests that you can use with or in place of this command or request.

Getting Help

If you have a technical question about ftServer system hardware or software, try these online resources first:

- **Online documentation at the StrataDOC Web site**. Stratus provides complimentary access to StrataDOC, an online-documentation service that enables you to view, search, download, and print customer documentation. You can access StrataDOC at the following Web site:

  http://stratadoc.stratus.com

- **Online support from Stratus Customer Service**. You can find the latest technical information about an ftServer system in the Stratus Customer Service Portal at the following Web site:

  http://www.stratus.com/go/support

  The Service Portal provides access to Knowledge Base articles for all Stratus product lines. You can locate articles by performing a simple or advanced keyword search, viewing recent articles or top FAQs, or browsing a product and category.

  To log in to the Service Portal, enter your employee user name and password or, if you have not been provided with a login account, click **Register Account**. When registering a new account, ensure that you specify an email address from a company that has a service agreement with Stratus.

If you cannot resolve your questions with these online self-help resources, and the ftServer system is covered by a service agreement, contact the Stratus Customer Assistance Center (CAC) or your authorized Stratus service representative. To contact the CAC, use the Service Portal to log a support request. Click **Customer Support** and **Add Issue**, and then complete the **Create Issue** form. A member of our Customer Service team will be glad to assist you.

Commenting on This Manual

You can comment on this manual using one of the following methods. When you submit a comment, be sure to provide the manual's name and part number, a description of the problem, and the location in the manual where the affected text appears.

- From StrataDOC, click the **site feedback** link at the bottom of any page. In the pop-up window, answer the questions and click **Submit**.
- From any email client, send email to `comments@stratus.com`.
- From the Stratus Customer Service Portal, log on to your account and create a new issue.

Stratus welcomes any corrections and suggestions for improving this manual.

Chapter 1
The System Directory on the Master Disk

The OpenVOS *installation software*, the software that is included with every new module, includes important system configuration files. Many of these files are stored in the `(master_disk)>system` directory (hereafter referred to as the system directory).

This chapter describes the contents of the system directory and outlines two procedures to follow when setting up your system. It includes the following sections.

The Master Disk

The *master disk* is the name of the member disk from which the module was booted. To identify the location of the master disk on an ftServer module, issue the command `display_disk_info (master_disk) -long`. This command displays the device ID of the master disk. You can use the device ID to determine the physical location of the disk.

The default master disk is the disk with a device ID of `0` (for example, `10/9/0/1/0`). This disk is not a physical disk; rather, it corresponds to the logical unit number (LUN) that has the device ID of `0`.

When you specify commands, you can reference the master disk with the command function `(master_disk)`. This is equivalent to the name of the top directory in that disk's directory hierarchy. Since some of the files in the system directory are used to boot the module, the master disk is also known as the *boot disk*. OpenVOS uses the `(master_disk)>Overseer` directory for both the home directory and the current directory during both an automatic boot and a manual boot.

For more information about master disks and boot disks, including information on the device ID of disks, see the manual *OpenVOS System Administration: Disk and Tape*

Administration (R284), which also provides information on using the `storage_monitor` command to monitor RAID controllers.

For information about supported PCI adapters, see the *Stratus ftServer Systems: PCI Adapter Guide* (R461).

The System Directory

Each master disk contains a system directory, which contains many subdirectories and files. The contents of this directory vary dramatically depending on the products for which your site has licenses.

This section describes the subdirectories and files your system directory **must** contain.

Table 1-1 lists the subdirectories required for the system directory.

Table 1-1. The Subdirectories in the System Directory

Subdirectories	
`accounting`	`message_library`
`command_library`	`object_library`
`configuration`†	`prom_code`
`doc`	`queues`†
`error`	`release_dir`†
`help`	`rsn_dir`†
`include_library`	`sample_programs`
`kernel_loadable_library`	`spooler`
`maint_library`	`transaction_logs`†

† Indicates that the directory is not system controlled. See the section "Controlled System Libraries" later in this chapter for more information.

- `accounting`

  Contains log files created by the accounting facility.

- `command_library`

  Contains all external commands shipped with OpenVOS.

- `configuration`

  Contains input data files for table files. (See the manual *OpenVOS System Administration: Configuring a System* (R287) for a definition of table files.) This directory also contains the `sample_module_start_up.`*release_number* file, which you customize to create the `module_start_up.cm` file for your module. This directory also contains `.table` files.

- `doc`

  Contains copies of previous and current software release bulletins (SRBs) and other online documentation. This directory also contains subdirectories for documentation on separately shipped products, as well as a series of files that have names in the form *`product_loaded`*`.memo`. Do not delete these product files.

- `error`

  Contains the files used as input to the error codes table. See Chapter 9, "Customizing Status Codes and Messages," for more information.

- `help`

  Contains a file of information that appears in response to the `debug` command's `help` request.

- `include_library`

  Contains include files to which the compilers and the assembler must have access.

- `kernel_loadable_library`

  Contains the program modules that are loaded using the `load_kernel_program` command. These include communications drivers, Forms Management System, and Stratus Office Solution.

- `maint_library`

  Contains tools used by the Stratus Customer Assistance Center (CAC) or your authorized Stratus service representative.

- `message_library`

  Contains per-command message files and help files. All Stratus-supplied files are in the subdirectory `us_english`.

- `object_library`

  Contains object modules referenced by the binder.

- `prom_code`

  Contains microcode for use by Stratus personnel only.

- `queues`

  Contains directories that hold queue files for the spooler, batch, and Remote Job Entry facilities.

- `release_dir`

  Contains files used during the installation of a new release of OpenVOS.

- `rsn_dir`

  Contains files received from the HUB of the Stratus Remote Service Network (RSN), which is described in Chapter 6, "The Remote Service Network." This directory is on one module within each system that is part of the RSN. This directory is useful only if your system has a Stratus maintenance contract.

- `sample_programs`

  Contains subdirectories of sample programs and any site-modifiable source for various parts of OpenVOS.

- `spooler`

  Contains printer and spooler-related files. The spooler directory contains the `spooler_configuration.v1.table` file and may contain site-specific printer information. For more information on this table, see the manual *VOS System Administration: Administering the Spooler Facility* (R286).

- `transaction_logs`

  Contains a log of all transactions and a work area for use in completing transactions.

Controlled System Libraries

The installation procedure imposes restrictions on the contents of most system libraries. (The libraries that are not system controlled are marked with a dagger (†) in Table 1-1.) The contents of these *controlled system libraries* are managed by the installation tools, and after the installation is complete, these libraries contain only those files supported in the particular release. This results in the automatic deletion of obsolete or renamed commands, documentation files, and so on.

NOTICE

If your site has added files or links to these libraries, they will be deleted by the installation process.

If you have stored user-written application programs or macros in controlled system libraries, you must create separate user libraries for these files and move them. You can set up local libraries by performing the following procedure.

1. Create a directory to be used for your local library; for example, `(master_disk)>system>applications_library`.
2. Edit the `module_start_up.cm` file to add the `add_default_library_path` and `add_library_path` commands for the directory `(master_disk)>system>applications_library`. (For a description of the `add_library_path` command, see the *OpenVOS Commands Reference Manual* (R098).)
3. Move user-written programs and macros into this library.

NOTE

The file `(master_disk)>system>error>user_codes.tin` is not deleted by the installation tools if the file exists at your site.

The Files in the System Directory

Table 1-2 lists the files required for the system directory.

Table 1-2. The Files in the System Directory

Files	
`abbreviations`	`Module_Config`
`backbone_systems.table`	`Module_Config.txt`
`batch.one_way_server_queue`	`module_start_up.cm`
`batch.server_queue`	`network_access.table`
`broadcast_queue.message_queue`	`network_routes`
`cc_log.`*`date`*	`new_backbone_systems.table`
`change_password.sysdb`	`new_modules.table`
`current_release`	`new_systems.table`
`devices.table`	`nodes.table`
`disks.table`	`notices`
`echo_event`	`overseer.message_queue`
`edit_help.table`	`overseer.server_queue`
`error_codes.text`	`release_tape_log`
`file_broadcaster.log`	`remote_maint_log.`*`date`*
`gateways.table`	`rsn_disk_threshold.table`
`hardware_error_event`	`security_log.`*`date`*
`hardware_log.`*`date`*	`sel_log.`*`date`*
`HUB.comm_MQ`	`software_purchase_table.table`
`HUB.comm_SQ`	`spooler_configuration.table`
`HUB.config`	`spooler_configuration.v1.table`
`HUB.disk_data`	`start_up.cm`
`HUB.msg_SQ`	`syserr_log.`*`date`*
`load_configuration.table`	`syserr_log_event`
`load_control_event`	`system_attributes.table`
`load_control_log.`*`date`*	`system_classes.table`
`load_control.server_queue`	`system_libraries.table`
`load_control.table`	`user_registration.sysdb`
`login_screen_image`	

Some of the `.table` and `.log` files are described in the "Table Files" and "Log Files" sections, respectively, later in this chapter. Here is a brief description of the other necessary files in the system directory.

- `abbreviations`

  Contains standard abbreviations for many OpenVOS commands. OpenVOS reads this file when the `use_abbreviations` command is executed (usually at user startup).

  When the `registration_admin` command creates a home directory for a new user, the command copies this file to the home directory. You can change the contents of this file to suit your system, and individual users can tailor their copies of the file to their own needs. See the *OpenVOS Commands User's Guide* (R089) for information about the `abbreviations` facility.

- `batch.one_way_server_queue`
  `batch.server_queue`

  Communicate with the Batch Overseer process. The Overseer uses `batch.one_way_server_queue` to communicate with the Batch Overseer; certain commands, such as `batch_admin`, use `batch.server_queue` for communications to and from the Batch Overseer. See Chapter 4, "The Batch Processor."

- `broadcast_queue.message_queue`

  Holds broadcast requests issued with the `broadcast_file` command. You can display the contents of this file with the `list_broadcast_requests` command, and you can delete a request from the file with the `cancel_broadcast_requests` command. See the manual *OpenVOS System Administration: Configuring a System* (R287) for more information about these commands.

- `change_password.sysdb`
  `user_registration.sysdb`

  Registration database files, created by the command `create_user_sysdbs`.

- `echo_event`
  `hardware_error_event`
  `syserr_log_event`

  Create empty files that provide file names for events that are notified in cases such as hardware errors.

- `error_codes.text`

  Contains information about all of the OpenVOS error codes. This file is created by the `update_codes` command. See Chapter 9, "Customizing Status Codes and Messages."

- `file_broadcaster.log`

  An output file created by the most recent execution of the `broadcast_file` command.

- `hardware_log.(date)`
  `sel_log.(date)`
  `syserr_log.(date)`

  Contain the messages that OpenVOS writes to the hardware log, sel log, and system error log files.

  System event log (sel log) files contain a record of all the SELs generated by the Baseboard Management Controller (BMC) on that day. If no SELs are generated on a particular day, no sel log exists for that day. Sel log files are created by

OpenVOS with implicit locking: once opened for input, the current day's sel log file cannot be deleted until it is closed at the end of the day.

System error log files are created by OpenVOS with implicit locking: once opened for input, the current day's system error log file cannot be deleted until it is closed at the end of the day.

Keep these log files for at least two weeks. If the CAC is investigating problems on your module, contact the CAC to verify that the information is no longer needed before deleting the files.

- `HUB.comm_MQ`
  `HUB.comm_SQ`
  `HUB.config`
  `HUB.disk_data`
  `HUB.msg_SQ`
  `remote_maint_log.`*`date`*

  Files used by the Remote Service Network (RSN), which is described in Chapter 6, "The Remote Service Network." Some of these files are on only one module within each system that is part of the RSN. The RSN files are useful only if your system has a Stratus maintenance contract.

  Keep remote maintenance log files for two weeks, and then delete them.

- `load_control_event`
  `load_control_log.(date)`
  `load_control.server_queue`

  These files and the load control table files are used by OpenVOS to measure the load on a module. See Chapter 5, "The Load Control Facility," for a description of the load control facility.

  Keep load control log files for two weeks, and then delete them.

  The load control facility creates a file whenever it starts a process to increase the load on the module. Information about the process is written to a file named `start_process.ila`*`N`*, which is placed in the system directory. In the file name, `ila` means "increase load action," and *`N`* is the count of these files that have been created on the module.

- `Module_Config`

  A machine-readable file that contains complete configuration information about the current module, used by the CAC.

- `Module_Config.txt`

  A text version of the *`Module_Config`* file, used by the CAC.

- `module_start_up.cm`

  Contains commands that OpenVOS uses at module startup. This file **must** be present in the system directory of every module.

  For protection against loss of the `module_start_up.cm` file, the installation software stores a standard copy of this file in the `(master_disk)>system>configuration` directory. Always keep a printed copy of the current version of this file, since your site may have modified it.

  See the manual *OpenVOS System Administration: Starting Up and Shutting Down a Module or System* (R282) for a detailed description of `module_start_up.cm`.

- `network_routes`

  Contains information about StrataNET routing for wide area network operation. This file is useful only if your system includes the StrataNET communications network. See the manual *VOS Communications Software: X.25 and StrataNET Administration* (R091) for information about wide area networks.

- `notices`

  An ASCII text file containing messages for module users. OpenVOS displays this file when the `display_notices` command is executed (usually at user startup). You maintain this file.

- `overseer.message_queue`

  Used by servers to communicate with each other.

- `overseer.server_queue`

  Used by certain commands (for example, `batch_admin`) for communication to and from the Overseer process.

- `security_log.`*`date`*

  Contains a record of each attempt by a user to access files and directories in violation of the system's access control. See the manual *OpenVOS System Administration: Registration and Security* (R283) for more information.

  Security log files are created by OpenVOS with implicit locking: once opened for input, the current day's security log file cannot be deleted until it is closed at the end of the day.

  Keep these log files for two weeks, and then delete them.

- `start_up.cm`

  A user command macro executed upon user login. The following example shows a simple `start_up.cm` file.

  ```
  use_abbreviations
  &if (process_type) = batch &then &return
  display_notices
  ```

  At login, the user process reads `start_up.cm` from the user's home directory. The `use_abbreviations` command tells OpenVOS to recognize the abbreviations in the `abbreviations` file in the user's home directory. The `display_notices` command displays on the user's terminal all files and all targets of links to files matching the star name `(master_disk)>system>notices*`.

  When the `registration_admin` command creates a home directory for a new user, the command copies this macro to the home directory. You can change the contents of the `start_up.cm` file to suit your system, and individual users can tailor their copies of the file to their own needs. For example, system administrators may want to include in their `start_up.cm` files the commands `notify_hardware_error`, `notify_security_violation`, and `rsn_mail`.

- `system_attributes.table`

  Contains information necessary to collect the module's hardware configuration information that is used by the CAC.

- `system_classes.table`

  Contains information necessary to collect the system attributes information used by the CAC.

Table Files

The system directory contains several files whose names have the suffix `.table`. These are *table files*.

Most table files are created by the `create_table` command from input files contained in the `configuration` directory. The following table files are exceptions.

- the `system_libraries.table` file is created by the installation procedure
- the *`rsn_disk_threshold.table`*, *`system_attributes.table`*, and *`sys`*`tem_classes.table` files are created by the CAC

See the manual *OpenVOS System Administration: Configuring a System* (R287) for information about these files.

The `new_modules.table`, `new_systems.table`, and `new_backbone_systems.table` files are used in conjunction with cross-module and cross-system communications products (for example, Open StrataLINK and X.25 StrataNET). For information about Open StrataLINK, see the *VOS Administrator's Guide for Open StrataLINK* (R388). For information about X.25 StrataNET, see the manual *VOS Communications Software: X.25 and StrataNET Administration* (R091).

Log Files

The system directory contains several files whose names have the suffix `.(date)`. These are *log files*.

For each type of log file, OpenVOS starts a new file each day when it writes the first entry after 12:00 a.m. The `.(date)` suffix of each file name is the date the log was written. For example:

```
hardware_log.21-11-01
```

If you specify a log file name with `.(date)` instead of the actual date, OpenVOS uses the current date. Keep log files for two weeks, and, unless they are being used to help debug a system problem, delete them.

Creating a Principal Master Disk

In a multimodule system, you can keep the `configuration` directory on the master disk of only one module. The master disk containing this directory is then known as the *principal master disk*. It should be on a fully duplex module. The module that contains the principal master disk should also have the MaintBridgeServer process.

To access the `configuration` directory from other modules, create a link from each of the other modules to the principal master disk. For information about creating links, see the description of the `link` command in the *OpenVOS Commands Reference Manual* (R098).

Perform the following steps to designate one master disk as the principal master disk.

1. Choose a master disk on a fully duplexed module to be the principal master disk.
2. Delete the `configuration` directory from every other module in the system.
3. In the directory `(master_disk)>system` on every other module, create links to the `configuration` directory in the `(master_disk)>system` directory on the module containing the principal master disk.

All references in this manual to the `configuration` directory assume that, if your system has more than one module, you have followed these steps to create a link to the principal master disk.

Storing Copies of Important Files

The file `error_codes.tin` is stored in the `error` directory and contains the text of OpenVOS error messages. Store a copy of this file, on a separate system or in hardcopy, in case you cannot access the online file. A hardcopy of the file may be helpful if problems occur during a manual boot. See Chapter 9, "Customizing Status Codes and Messages," for more information about this file.

Also, store a copy of each module's `module_start_up.cm` file, in case you lose both copies of it.

Chapter 2

Setting and Defining Priority Levels

Stratus modules support two types of processes: interactive and batch processes. *Interactive processes* are started with the `login` command. *Batch processes* are started with the `batch` command, the `start_process` command, or the `s$start_process` subroutine.

OpenVOS schedules CPU time for each interactive and batch process running on a module based on the following criteria: the priority level of each process and the parameters that define the meaning of the priority levels. There are 10 priority levels, 9 being the highest and 0 the lowest. Note, however, that since OpenVOS schedules a process for each CPU as the CPU becomes available, process priorities are only significant when there are more processes ready to run than there are CPUs.

This chapter explains how to set and define priority levels. It contains the following sections.

- "Setting Priority Levels" on page 2-1
- "Defining Priority Levels" on page 2-2

Setting Priority Levels

The priority of an interactive process, if you do not specify a different priority, is your *default priority*. It is set in the `Priority` field of your entry in the registration database. If your entry does not contain a value for this field, OpenVOS assigns a default priority of `0`.

The registration database also can assign a *maximum priority*, which limits how high a priority can be on a process that you start. This value is assigned in the `Max Priority` field. If no value is assigned, OpenVOS assigns a maximum priority of `0`.

Whenever you start a process, you can assign a priority to the process. There are three commands (`login`, `batch`, and `start_process`) and one subroutine (`s$start_process`) with which you can start a process, and each of these has an argument for setting the priority of the process. (The priority cannot be higher than your maximum priority.) When you do not assign a priority for a new interactive process, OpenVOS assigns your default priority, and for a new batch process, OpenVOS assigns the priority of the calling process.

Changing the Priority of a Process

Once a process is started, you can change its priority by issuing the `set_priority` command. In the command, if you are privileged, you can specify one or more processes by providing values for the arguments *`process_name`* and `-user` *`user_name`*. Both arguments accept star names. If you do not specify any processes, OpenVOS changes the priority of the process issuing the command.

You can make the following changes in the priority of a process by issuing the `set_priority` command.

- change the priority of another user's process
- assign to a process a higher priority than the maximum priority of the user who owns the process—This applies to all processes, including your own.

The following command assigns a priority of 7 to the user Smith's process named `make_report` on the current module.

```
set_priority  7  make_report  -user Smith
```

The following command assigns a priority of 6 to all of your processes on module `m7`.

```
set_priority  6  *  -module m7
```

For a description of the `set_priority` command, see *OpenVOS Commands Reference Manual* (R098).

Defining Priority Levels

OpenVOS has default values for scheduling parameters, but you can also define priority with the `set_scheduler_info` command. Chapter 14 describes this command.

The following sections discuss defining priority levels.

- "Time Slices" on page 2-3
- "Network Sockets" on page 2-5
- "OpenVOS Default Scheduling Parameters" on page 2-6
- "Changing Scheduling Parameters" on page 2-9

Time Slices

A *time slice* is an interval of CPU time that OpenVOS allocates to a process. The amount of CPU time that a process receives depends on the attributes of the time slices assigned to that priority level and process type. These attributes are built into the installation software, and they can be changed, as described in ‘‘Changing Scheduling Parameters’’ on page 2-9.

Time slices have the following attributes:

- `type`

  Specifies the type, represented by a number between 1 and 255. The `type` number determines the relative priority, with a type-1 time slice having the highest priority and a type-255 time slice having the lowest priority.

  NOTE

  For consistency in scheduling and optimal performance, the processes with the highest priority number (9, 8, and 7) should use time slices with the lowest type number (2, 4, and 6, by convention), which ensures that essential kernel processes are given appropriate access to the CPUs, in order to complete their work in a timely way.

- `count`

  Specifies the number of time slices of the type specified that a process can receive. The range is from 1 to 16.

- `time`

  Specifies the number of seconds that OpenVOS allocates for each time slice of the type specified. The range of allowed values is 0.002 to 4.000, in seconds.

When you assign a time-slice type number, you must assign count and time values for that type.

Assign values for these attributes using the `set_scheduler_info` command, as described in ‘‘Changing Scheduling Parameters’’ on page 2-9. The `display_scheduler_info` command displays the attributes presently assigned.

For scheduling purposes, processes are ordered by their time-slice type numbers, beginning with the lowest type number. OpenVOS does not include the time spent waiting for I/O in a process’s time slice.

Each priority level/process type can have up to four time-slice type numbers, with corresponding count and time values assigned to it. Each time a process uses all of the time slices specified by a `count` value, it switches to the next time-slice type number. This continues until the process reaches the last of the time-slice type numbers, where it continues to process regardless of the specified `count`. When a process finishes processing and enters *wait mode*, OpenVOS returns the process to the first time-slice type number specified for that process priority.

NOTE

Pre-emption is the default. Several times per second, the scheduler interrupts running processes to determine if a more eligible process should run.

The following example defines the time slices assigned to batch processes with a priority of 0. The `SKT` (socket) field represents an address on a module used for network communications and is explained in ''Network Sockets'' on page 2-5.

```
        -----BATCH-----
   PRI    SKT  TYPE  CNT  TIME
    0       1   14    1   0.250
```

In this example, OpenVOS allocates the batch process a single type-14 time slice that is 0.250 seconds in length.

The following example defines the time slices assigned to any interactive process with a priority of 2.

```
          --INTERACTIVE--
   PRI     SKT  TYPE  CNT  TIME
    2        1     2    1  0.063
                   6    5  0.250
                   6    5  0.125
                  10    1  0.125
```

For example, assume that the process `compile_reports` has a priority of 2. OpenVOS allocates CPU time to the process using the time slice definitions displayed in the previous example. The following sequence describes this allocation.

1. When `compile_reports` is invoked, OpenVOS runs it as soon as all other processes with type-2 time slices use up their time. If no other type-2 time slices or no processes with lower numbered time-slice types are competing for time, OpenVOS runs `compile_reports` for one time slice of 0.063 seconds.

   OpenVOS then allocates to the `compile_reports` process its next numbered time-slice type, as explained in the next step.

2. The next time-slice type allocated to the `compile_reports` process is a type-6 time slice that is 0.250 seconds long. The time slice of type 6 lowers the scheduling priority of `compile_reports`, because it can now be pre-empted by processes

with time slices of type 1, 2, 3, 4, or 5. OpenVOS can allocate up to five of these type-6, 0.250-second time slices to `compile_reports`.

When the five type-6, 0.250-second slices are used up, OpenVOS allocates to the `compile_reports` process its next numbered time-slice type, as explained in the next step.

3. The third time-slice type allocated to the `compile_reports` process is a type-6 time slice that is 0.125 seconds long. OpenVOS can allocate up to five of these type-6, 0.125-second time slices to `compile_reports`.

   When the five type-6, 0.125-second slices are used up, OpenVOS allocates to the `compile_reports` process its next numbered time-slice type, as explained in the next step.

4. The last time-slice type allocated to the `compile_reports` process is one type-10 time slice that is 0.125 seconds long. Since this is the last entry in the list, the process receives this time-slice type until it enters wait mode.

Whenever the process waits for any reason, its scheduling parameters are reset to the first entry in the list.

Network Sockets

A *network socket* is an OpenVOS addressing mechanism used for Open StrataLINK and/or StrataNET communication. ftServer modules support Open StrataLINK communication.

NOTE

Sockets in this context are not the same as TCP/IP sockets.

There are two types of network sockets: reserved Sockets, which are assigned during the configuration of StrataLINK or StrataNET, and internally-assigned network sockets, which are used by network processes.

Each StrataNET network client process is assigned a unique socket number to reserve that socket for a specific type of communication. For example, you associate an StrataNET network client process with a socket number using the `-socket` argument of the `network_client` command. There are conventions for assigning socket numbers for the StrataNET network client process (for example, socket 11). For more information, see the manual *VOS Communications Software: X.25 and StrataNET Administration* (R091).

You can also associate the socket reserved for a networking process with a specific priority level. To associate a specific socket with a priority level, issue the `set_scheduler_info` command, which is described in Chapter 14.

For any networking configuration that includes Open StrataLINK and/or StrataNET, you should determine how many sockets and networking processes you need and how to efficiently handle the processes at the appropriate priority levels. The `module_start_up.cm` file shipped with each release shows sample command lines for the different types of networking processes. Note that the current implementation of Open StrataLINK does not use OpenVOS reserved sockets.

OpenVOS Default Scheduling Parameters

This section describes how the initial process scheduling values affect system performance, explains why some OpenVOS processes have certain priorities, and provides recommendations about assigning priorities to processes within the current scheduling plan.

For each priority level (0-9), in both types of processes (interactive and batch), the assigned default values define the attributes of the time slices that processes can receive. OpenVOS uses these values to compute the amount of CPU time it schedules for processes at each priority level. The following output shows these values as displayed by the `display_scheduler_info` command.

```
--INTERACTIVE--             -----BATCH-----
PRI    SK

YPE  CNT  TIME         SKT  TYPE  CNT  TIME
 0       1    4    3   0.125        1   14     1   0.250
              6    5   0.250
              6    5   0.125
             10    1   0.125
 1       1    4    5   0.250        1   12     1   0.125
              4    5   0.125
              6    5   0.125
             10    1   0.125
 2       1    2    1   0.063        1   12     1   0.250
              6    5   0.250
              6    5   0.125
             10    1   0.125
 3       1    2    1   0.125        1   10     1   0.125
              4    3   0.250
              6    5   0.250
             10    1   0.125
 4       1    2    1   0.125        1   10     1   0.250
              4    3   0.250
              6    8   0.250
             10    1   0.125
 5       1    2    1   0.125        1   10     1   0.375
```

(Continued on next page)

```
             4    5   0.250
             6    5   0.250
            10    1   0.125
6       1    2    1   0.125          1    8    1   0.250
             4    5   0.250
             6    5   0.375
             8    1   0.125
7       1    2    1   0.125          1    6    1   0.125
             4    5   0.375
             6    5   0.375
             8    1   0.125
8       1    2    2   0.125          1    4    1   0.250
             4    5   0.375
             6    5   0.375
             8    1   0.125
9       1    2    3   0.125          1    2    1   0.250
             4    5   0.500
             6    5   0.500
             6    1   0.250
```

These scheduling parameters are designed to give higher priority to interactive processes than to batch processes. They are set so that interactive users will get quick responses.

Interactive-Process Scheduling Priorities

The following list describes the interactive-process scheduling priorities.

- Priorities 9, 8, and 7

  Available for processes that need very fast response. System performance may be degraded if these priorities are overused.

- Priority 6

  Recommended for processes that require a lot of CPU time, without waiting for the terminal, and that must run at a higher priority than normal batch jobs. After about 21 seconds of CPU time without waiting for the terminal, priorities 5, 4, and 3 degrade to a priority slightly lower than that of batch processes, while priority 6 degrades to a priority that will still pre-empt batch processes.

- Priority 5

  Recommended for normal interactive processes. By default, OpenVOS assigns priority 5 to prelogin processes. Contact the CAC if you want to change the default priority of the prelogin processes.

- Priorities 4 and 3

  Can be used, like priority 5, for normal interactive processes. The difference among priorities 3, 4, and 5 is small but noticeable.

- Priorities 2, 1, and 0

  Very low login priorities that should be used for processes running nonessential work. Priority 1 is noticeably below priority 2, and priority 0 is slightly below priority 1.

Batch-Process Scheduling Priorities

The following list describes the batch-process scheduling priorities.

- Priority 9

  Reserved for OpenVOS processes that need quick access to the CPU for short periods of time. These processes are, for example, the transaction processing Overseer (TPOverseer).

- Priority 8

  Reserved for OpenVOS processes that need to pre-empt all batch processes, except those with a priority of 9. The Overseer process is assigned to this priority.

- Priority 7

  Used by OpenVOS for system processes that need to run at a priority above the normal batch processes and above normal interactive processes that use a lot of CPU time. The BatchOverseer and the diagnostic utility process are assigned to this priority.

- Priority 6

  Not used for any OpenVOS process. Pre-empts normal batch jobs, but can be pre-empted by the priority 7 batch processes.

- Priorities 5, 4, and 3

  Recommended for normal batch processes. The difference among priorities 5, 4, and 3 is small but noticeable.

- Priority 2

  Recommended for processes that need to run only when the system is relatively idle. Runs at a priority below all interactive processes and shares time with priority 1 batch processes.

- Priority 1

  Used by OpenVOS for disk recovery. Shares resources with priority 2 batch processes and pre-empts priority 0 batch processes. Runs at a priority below all interactive processes.

- Priority 0

  Available for `start_process` processes that are to run only when the module is otherwise completely idle. Will be pre-empted by all other priorities.

Changing Scheduling Parameters

The `set_scheduler_info` command sets, for the current bootload, the parameters OpenVOS uses when scheduling processing time. The command sets the parameters for one priority level of one type of process, assigning from one to four time-slice types to the specified process type/priority level. Every time-slice type value specified in the command must have corresponding count and time values.

You can change the scheduling parameters. For example, you can assign parameters so that 0 is the highest priority, 5 is the lowest, and 6 through 9 are unused. Since each process type (batch and interactive) has 10 priority levels (0-9), changing the full set of scheduling parameters requires that you issue the `set_scheduler_info` command 20 times.

If you want the changed parameters to be effective beyond the current bootload, include `set_scheduler_info` in the `module_start_up.cm` file.

The following command sets the scheduling parameters for interactive processes with a priority of 8 on module `m2`.

```
set_scheduler_info  8  -type 1 2 4  -count 1 5 1
     -time 1000 2000 4000  -module m2
```

After the preceding command is executed, the `display_scheduler_info` display for module `m2` is as follows:

```
--INTERACTIVE--
PRI   SKT  TYPE  CNT  TIME
 8      1    1    1   1.000
             2    5   2.000
             4    1   4.000
```

The following command sets the scheduling parameters for batch processes at priority level 6 on the current module.

```
set_scheduler_info  6  -type 7 5  -count 1 2  -time 2000 1000
     -process batch
```

After the preceding command is executed, the `display_scheduler_info` display is as follows:

```
     -----BATCH-----
PRI   SKT  TYPE  CNT   TIME
 6     1    7     1    2.000
            5     2    1.000
```

Chapter 3
Setting a Module's Default Command Limits

A module's *default command limits* specify the initial and maximum command limits for each process running on a Stratus module. These default command limits, in conjunction with per-process command limits, determine how much heap, stack, and total program module memory space is available to each process.

You set a module's default command limits with the `update_default_cmd_limits` command and list these defaults with the `list_default_cmd_limits` command. The `update_default_cmd_limits` and `list_default_cmd_limits` commands are described in detail in Chapter 14.

You can set the per-process command limits with the `update_process_cmd_limits` command and the `s$set_current_cmd_limit` subroutine. You can list these per-process limits with the `list_default_cmd_limits` command or `s$get_current_cmd_limit` subroutine. For more information on the `update_process_cmd_limits` command, see the *OpenVOS Commands Reference Manual* (R098). For more information on the subroutines, see the OpenVOS Subroutines manuals.

This chapter describes how to set a module's default command limits. It contains the following sections.

- "Setting the Default Heap Limits" on page 3-1
- "Setting the Default Stack Limits" on page 3-2
- "Setting the Default Total Limits" on page 3-3

Setting the Default Heap Limits

You can specify, in bytes, the default initial and maximum heap limits for commands started on the specified module using the `update_default_cmd_limits` command. A *heap* is an area in virtual address space where OpenVOS allocates dynamic variables using either the `s$allocate` subroutine or language-specific memory allocation statements. Space in a heap must be allocated and freed dynamically.

Use the following rules to set the values of the `-initial_heap_limit` and `-maximum_heap_limit` arguments of the `update_default_cmd_limits` command. Programmers may want the default limits changed to accommodate an application's memory space requirements.

- The value that you specify for the `-initial_heap_limit` argument must be less than or equal to the value of the `-maximum_heap_limit` argument and 32,768 bytes or greater. The default value is `infinity` bytes for ftServer modules.
- The value that you specify for the `-maximum_heap_limit` argument must be greater than or equal to the value of the `-initial_heap_limit` argument and less than or equal to the value you specify in the `-maximum_total_limit` argument. The default value is `infinity` bytes for ftServer modules.

Since commands and program modules in privileged processes can exceed the `-maximum_heap_limit` value, you may want to check who owns privileged processes on your module.

Setting the Default Stack Limits

You can specify, in bytes, the initial and maximum stack limits for commands started on the specified module. A *stack* is an area of virtual address space consisting of an ordered series of stack frames associated with the execution of a program.

Use the following rules to set the values of the `-initial_stack_limit` and `-maximum_stack_limit` arguments of the `update_default_cmd_limits` command. Programmers may want the default limits changed to accommodate an application's memory space requirements.

- The value that you specify for the `-initial_stack_limit` argument must be less than or equal to the value of the `-maximum_stack_limit` argument and 32,768 bytes or greater. The default value is 8,388,608 bytes for ftServer modules.
- The value that you specify for the `-maximum_stack_limit` argument must be greater than or equal to the value of the `-initial_stack_limit` argument and less than or equal to the value you specify in the `-maximum_total_limit` argument. The default value is 134,217,728 bytes for ftServer modules.

Since commands and program modules in privileged processes can exceed the `-maximum_stack_limit` value, you may want to check who owns privileged processes on your module.

Setting the Default Total Limits

You can specify, in bytes, the default initial and maximum total limits for commands started on the specified module. Total memory space includes a command's stack, heap, shared and unshared static regions, and code region. The following list defines shared static, unshared static, and code regions.

- A *shared static region* contains external static data that can be shared by several tasks.
- An *unshared static region* contains internal static (temporary per-process) data that can be referenced by only one task in the program.
- A *code region* contains a program's machine code instructions.

NOTE

The size of the shared static, unshared static, and code regions are determined by the binder. Use the `display_program_module` command to learn the size of these regions.

Use the following rules to set the values of the `-initial_total_limit` and `-maximum_total_limit` arguments of the `update_default_cmd_limits` command. Programmers may want the default limits changed to accommodate an application's memory space requirements.

- The value that you specify for the `-initial_total_limit` argument must be less than or equal to the value of the `-maximum_total_limit` argument and 131,072 bytes or greater. The default value is `infinity` for ftServer modules.
- The value that you specify for the `-maximum_total_limit` argument must be greater than or equal to the value of the `-initial_total_limit` argument. The default value is `infinity` for ftServer modules.

Since commands and program modules in privileged processes can exceed the `-maximum_total_limit` value, you may want to check who owns privileged processes on your module.

Chapter 4
The Batch Processor

The *batch processor* is an OpenVOS facility that creates processes so that users can execute commands independently of their interactive processes. This chapter describes how to administer the batch processor to ensure that it is operating efficiently and is receiving sufficient system resources. You use the following commands to administer batch processing.

- `batch_admin`
- `create_batch_queue`

These commands are described in more detail in Chapter 14.

This chapter contains the following sections.

- "Batch Queues" on page 4-2
- "Managing Batch Traffic" on page 4-6

You can work with the load control program to control batch traffic, as described in Chapter 5, "The Load Control Facility."

The end-user batch commands, documented in the *OpenVOS Commands Reference Manual* (R098), are as follows:

- `batch`
- `cancel_batch_requests`
- `display_batch_status`
- `list_batch_requests`

Batch Queues

When you submit a batch request, it is placed in a *batch queue*, which is a file that holds batch requests while they await processing. Batch queues are located in the following directory.

```
(master_disk)>system>queues>batch
```

To create batch queues, use the `create_batch_queue` command, as described in the section "Creating Batch Queues" later this chapter. You can create batch queues on any module, and the number of batch queues that can exist on a module is unlimited.

Each batch queue is controlled by one or more BatchOverseer processes, as designated by the `-overseer_module` argument to the `batch_admin` command. (BatchOverseer processes on a given module are started and managed by the Overseer process on that module.) The BatchOverseer on any module can be responsible for multiple queues, any of which can be located on another module. Furthermore, BatchOverseer processes on any number of modules can be responsible for the same batch queue.

The `-queue` and `-module` arguments to the `batch` command allow a user to specify the queue to which a batch request is submitted.

The following sections describe how to administer batch queues.

- "Starting the `normal` Batch Queue" on page 4-2
- "Creating Batch Queues" on page 4-3
- "Restricting Access Rights to Batch Queues" on page 4-3
- "Deleting Batch Queues" on page 4-3
- "Starting a Batch Queue" on page 4-4
- "Stopping a Batch Queue" on page 4-5

Starting the `normal` Batch Queue

The default batch queue is known as the `normal` batch queue. To start the default batch queue during module startup, the installation software uses the load control facility (described in the section "Automatic Management (Load Control)" later in this chapter).

If the `normal` queue is lost or inadvertently deleted, you can create a new queue called `normal` with the `create_batch_queue` command.

Creating Batch Queues

To create a batch queue, issue the `create_batch_queue` command and supply a name for both the queue you are creating and the module that will contain the queue. If you do not specify a module name, OpenVOS creates the queue on your current module. OpenVOS always stores the queue in the `(master_disk)>system>queues>batch` directory.

For example, the following command creates a batch queue called `compute` on module `m3`.

```
create_batch_queue  compute  -module m3
```

NOTE

If you create a batch queue other than `normal`, you **must** include the `batch_admin` `start` request in the `module_start_up.cm` file.

Restricting Access Rights to Batch Queues

A user who has read or write access to a batch queue can see a list of all requests in the queue by issuing the `list_batch_requests` command with the `-all` argument. However, a user who has execute access to the queue can only see his or her own batch requests by issuing this command. So, to restrict a user's access to information about other users' requests in a batch queue, give that user execute access to the queue.

For more information about access rights, see the manual *OpenVOS System Administration: Registration and Security* (R283).

Deleting Batch Queues

To delete a batch queue from the directory `(master_disk)>system>queues>batch`, issue the `delete_file` command, specifying the name of the queue for the *`file_names`* argument. You can delete more than one queue at a time with this command.

The *OpenVOS Commands Reference Manual* (R098) documents the `delete_file` command.

The following command sequence deletes the batch queues `batch_1` and `batch_2`.

```
change_current_dir  (master_disk)>system>queues>batch
delete_file  batch_1  batch_2
```

Starting a Batch Queue

To start processing requests from a specified queue, issue the following command.

```
batch_admin  start  queue_name
```

If the queue is located on another module, you must name that module in the `-module` *`queue_module`* argument. If you want a module other than the current module to run the batch processes in the specified queue, supply the name of that module in the `-overseer_module` *`overseer_module`* argument.

If you want to specify the maximum number of batch requests that the batch overseer module can process at one time from the specified queue, supply the number in the `-max_users` *`number`* argument. The default value for *`number`* is `1`.

The installation software uses the load control facility (discussed in the section "Automatic Management (Load Control)" later in this chapter) to start the `normal` batch queue during module startup.

NOTE

After a module reboot, an error message may be returned if all system servers are not yet functional when you issue the `batch_admin` command to restart the queues. For example, the system could return a message that the queue has no message server if the Overseer process has not yet reached full functionality after the reboot. If this occurs, wait a few minutes and then reissue the `batch_admin` command.

The following command tells OpenVOS on the current module to start processing a maximum of one request at a time from the batch queue `compute`, located on module `m8`.

```
batch_admin  start  compute  -module m8
```

The following command tells OpenVOS on module `m10` to start processing a maximum of three requests at a time from the batch queue `normal`, located on the current module.

```
batch_admin  start  normal  -overseer_module m10  -max_users 3
```

The following command tells OpenVOS on module `m3` to start processing a maximum of one request at a time from the batch queue `batch_1`, located on module `m6`.

```
batch_admin  start  batch_1  -module m6  -overseer_module m3
```

Stopping a Batch Queue

To stop processing requests from a specified queue, issue the following command.

```
batch_admin  stop  queue_name
```

If the queue is located on another module, you must name that module in the `-module` *`queue_module`* argument.

If the batch processes on the specified queue are being run on a module other than the current module, you must name it in the `-overseer_module` *`overseer_module`* argument. The command `batch_admin stop_all -overseer_module` *`overseer_module`* stops processing requests from all batch queues being processed by *`overseer_module`*.

When the batch processor receives a `stop` request, it processes the current request and then stops. To stop the current request before it is processed, use either the `cancel_batch_requests` command or the `stop_process` command. You can also use the `cancel_batch_requests` command to remove a request from a batch queue before the batch processor begins to process it.

The following command stops batch processing on the current module from the `normal` queue, located on the current module.

```
batch_admin  stop  normal
```

The following command stops batch processing on the current module from the queue `batch_1`, located on module `m6`.

```
batch_admin  stop  batch_1  -module m6
```

The following command stops batch processing on module `m9` from the queue `batch_2`, located on the current module.

```
batch_admin  stop  batch_2  -overseer_module m9
```

The following command stops batch processing from all batch queues being processed on module `m9`.

```
batch_admin stop_all -overseer_module m9
```

Managing Batch Traffic

Batch traffic is managed automatically by the load control facility, but you can use the `batch_admin` command to manage batch traffic manually. This section describes how to manage batch traffic manually. It contains the following sections.

- "Automatic Management (Load Control)" on page 4-6
- "Manual Intervention" on page 4-6

Automatic Management (Load Control)

The load control facility, described in Chapter 5, helps to manage batch processing. When the load on a module becomes heavy, as defined by a load control table, the load control facility tells the BatchOverseer on that module to stop processing batch requests from the `normal` queue that is run by that module. When the load control program finds that the load has become lighter, the BatchOverseer resumes processing batch requests from the `normal` queue.

You can modify the load control tables so that other queues on a module are automatically stopped and started according to the module's processing load.

Manual Intervention

In addition to adjusting the load control facility to suit your system's needs, there are several other ways you can monitor and adjust the amount of system resources claimed by the batch processor.

To monitor batch processing, use the following commands to display information about the traffic the batch processor is handling.

```
list_batch_requests
display_batch_status
```

To adjust batch traffic, perform the following steps.

1. Specify the `stop` request of the `batch_admin` command. This stops batch processing on one or more queues to free system resources for other work.
2. Specify the `-overseer_module` *`overseer_module`* argument of the `batch_admin` command to assign a queue to a different module. This relieves a module of its responsibility for running one or more batch queues.
3. Specify the `-max_users` *`number`* argument of the `batch_admin` command. This restricts the number of batch processes allowed to run concurrently from a specified queue.

You may find that independent processes already started with the `start_process` command are using system resources so heavily that the batch processor cannot operate efficiently. In this case, update the registration database entries for some or all users to reduce the number of processes that they are allowed to create. To update registration database entries, issue the `registration_admin` command, which invokes a menu-driven set of screens. You can specify any number from 0 to 255 in the `Max Processes` field of the third `UPDATE USER INFORMATION` screen for a user. A nonzero value limits the user to the specified number of processes. (Specifying `0` is the same as specifying `255`.) For more information, see the `registration_admin` command description in the manual *OpenVOS System Administration: Registration and Security* (R283).

Chapter 5
The Load Control Facility

OpenVOS's dynamic *load control facility* monitors the processing load automatically on each module of a system and initiates actions to increase or decrease the load. The facility includes the following components.

- a *load control process* on each module
- two *load control tables* that describe the parameters for the facility to use and actions for it to perform
- the capability for OpenVOS to keep records of all load control calculations and to convert these records into a histogram

You can start and stop a load control process at any time, and you can change the load control tables to suit the needs of any module in your system.

This chapter describes the load control facility. It contains the following sections.

Calculating the Load

The load control facility performs the following steps to calculate the load on a module.

1. It calls the subroutine `s$get_module_usage` and uses information from the `get_module_usage_info` structure to determine the *preliminary load factor* (the percentage of time the CPU is busy).
2. It collects a number of preliminary load factors and calculates their average. The number of factors collected is determined by the `repetition` field in the load control configuration table (described in the section "The `load_configuration.table` File" later in this chapter). The result is the *decisive load factor*.

3. It compares the decisive load factor to the *threshold values*, which are specified in the `load_configuration.table` file and referenced in the `load_control.table` file (described in the section "The `load_control.table` File" later in this chapter). If necessary, load control takes the action mandated in the action table to reduce or increase the load on the module.

Setting the Threshold Values

The load control facility has four *thresholds*. It compares these thresholds with the decisive load factors to determine if action is necessary to decrease or increase the load on a module. The thresholds are named as follows:

- `heavy`
- `medium_heavy`
- `medium`
- `light`

In the load control configuration table, you assign values to these names. In the load control action table, you reference these names, and OpenVOS uses the corresponding values to trigger the action specified in the load control action table to reduce or increase the load. The next section describes both tables.

The threshold values are integers between 1 and 100. The value 1 indicates the lightest possible load on the module; the value 100 indicates the heaviest load.

For example, the `load_configuration.table` file that comes with the installation software sets the following threshold values.

```
heavy          75
medium_heavy   60
medium         45
light          30
```

Use these values only if they seem appropriate for your system. Otherwise, change them according to your observations of how the load control facility functions on your system.

One criterion for deciding threshold values is your perception of response time on a module in relation to the load control facility's judgment of the load on that module. For example, if the load control facility consistently finds that the load on a module is heavy when, in your estimation, the response time for users is adequate, raise the value for the `heavy` threshold to a figure closer to 100.

Also consider the range between threshold values. If the load control facility is too frequently taking action to decrease and increase the load, change the threshold values that trigger these actions to create a wider range between them.

You can create a histogram of load control data that will help you determine if the threshold values on a module are appropriate. The section "Creating a Load Control Histogram" later in this chapter explains how to create a histogram.

The Load Control Tables

The load control facility includes two table files: `load_configuration.table` and `load_control.table`. The `load_configuration.table` file defines specific load information for each module, and the `load_control.table` file defines actions OpenVOS takes when the module load reaches a specified level.

The following sections describe the `load_configuration.table` and `load_control.table` files.

- "The `load_configuration.table` File" on page 5-3
- "The `load_control.table` File" on page 5-5
- "Load Control Tables in the Installation Software" on page 5-7
- "Changing a Load Control Table" on page 5-8

For general information about table files, see the manual *OpenVOS System Administration: Configuring a System* (R287).

The `load_configuration.table` File

In the `load_configuration.table` file, you define the following elements for each module in a system.

- the parameters that OpenVOS on that module uses to calculate the average load on the module—These are the `time_period` and `repetition` values.
- the values assigned to the load control threshold names

The `load_configuration.table` file contains one entry for each module in a system, and every module in a system has an identical copy of this table.

The `load_configuration.dd` File

The file `load_configuration.dd`, which you must **never** modify, appears as follows:

```
organization:  sequential;
index:         module_name no_duplicates;

fields: module_name      char(32) var,
        time_period      binary (31),
        repetition       binary (15),
        heavy            binary (15),
        medium_heavy     binary (15),
        medium           binary (15),
        light            binary (15);
end;
```

The Fields in the `load_configuration.table` File

The fields in the `load_configuration.table` file have the following meanings.

- `module_name` **Required**
  Specifies the name of the module to which the parameters in this entry apply.

- `time_period` **Required**
  Specifies the number of minutes OpenVOS waits before checking the module load.

- `repetition` **Required**
  Specifies the number of preliminary load factors OpenVOS collects before calculating their average and producing a decisive load factor.

- `heavy` **Required**
  Specifies a threshold value OpenVOS uses to indicate heavy module usage.

- `medium_heavy` **Required**
  Specifies a threshold value OpenVOS uses to indicate medium-heavy module usage.

- `medium` **Required**
  Specifies a threshold value OpenVOS uses to indicate medium module usage.

- `light` **Required**
  Specifies a threshold value OpenVOS uses to indicate light module usage.

Sample `load_configuration.tin` File Entries

The following example shows a sample `load_configuration.tin` file.

```
/    =module_name   m1
     =time_period   1
     =repetition    5
     =heavy         90
     =medium_heavy  70
     =medium        50
     =light         30

/    =module_name   m2
     =time_period   3
     =repetition    8
     =heavy         75
     =medium_heavy  60
     =medium        40
     =light         25

/    =module_name   m3
     =time_period   4
     =repetition    5
     =heavy         85
     =medium_heavy  55
     =medium        40
     =light         15
```

The `load_control.table` File

The `load_control.table` file defines the following elements for each module in a system.

- the action OpenVOS takes to reduce the load on a module when the load on that module exceeds a given threshold value
- the action OpenVOS takes to increase the load on a module when the load on that module decreases to a given threshold value

The table may have more than one entry for each module, defining actions to be taken for different thresholds on that module.

Every module in a system has an identical copy of this table.

The `load_control.dd` File

The following example shows the file `load_control.dd`, which you must **never** modify. (In the fields, `rla` stands for “reduce load action” and `ila` stands for “increase load action.”)

```
organization:  sequential;
index:         module_name duplicates;

fields:   module_name         char(32) var,
          rla_threshold       char(32) var,
          rla_command_line    char(256) var,
          ila_threshold       char(32) var,
          ila_command_line    char(256) var;
end;
```

The Fields in the `load_control.table` File

The fields in the `load_control.table` file have the following meanings.

- `module_name` **Required**
  Specifies the name of the module to which the actions defined in this entry apply.

- `rla_threshold`
  Specifies a threshold name for the threshold that is to trigger a decrease in the load.

- `rla_command_line`
  Specifies the action OpenVOS takes if the load on the module equals or exceeds the value of the threshold referenced in the `rla_threshold` field. This should be the opposite action from that specified in the `ila_command_line` field.

- `ila_threshold`
  Specifies a threshold name for the threshold that is to trigger an increase in the load.

- `ila_command_line`
  Specifies the action OpenVOS takes if the load on the module is less than or equal to the threshold value referenced in the `ila_threshold` field. This should be the opposite action from that given in the `rla_command_line` field.

  OpenVOS increases the load on a module by starting a process to perform the action specified in the `ila_command_line` field. Each time OpenVOS increases the load, OpenVOS also creates a file in the `(master_disk)>system` directory to hold information about the process. Each of these files has the name `start_process.ila`*N*, where *N* is the number of files created on the module.

The `ila_command_line` and `rla_command_line` fields can reference command files that you have created containing commands that will serve to reduce or increase the load. The following example references two such files, named `reduce_load.cm` and `increase_load.cm`.

Sample `load_control.tin` File Entries

The following example shows a `load_control.tin` file.

```
/     =module_name        m3
      =rla_threshold      heavy
      =rla_command_line   batch_admin  stop  normal
      =ila_threshold      medium
      =ila_command_line   batch_admin  start  normal

/     =module_name        m3
      =rla_threshold      medium_heavy
      =rla_command_line   batch_admin  stop  compute
      =ila_threshold      medium
      =ila_command_line   batch_admin  start  compute

/     =module_name        m5
      =rla_threshold      heavy
      =rla_command_line   reduce_load.cm
      =ila_threshold      medium_heavy
      =ila_command_line   increase_load.cm

/     =module_name        m5
      =rla_threshold      medium_heavy
      =rla_command_line   login_admin -restrict *.backup
      =ila_threshold      medium
      =ila_command_line   login_admin -unrestrict *.backup

/     =module_name        m6
      =rla_threshold      heavy
      =rla_command_line   batch_admin  stop  normal
      =ila_threshold      medium
      =ila_command_line   batch_admin start normal
```

Load Control Tables in the Installation Software

The installation software contains these files.

- the two `.dd` files shown in the sections "The `load_configuration.dd` File" and "The `load_control.dd` File" earlier in this chapter—**Never** modify these files.

- a `load_configuration.tin` file that contains an entry for each module in your system—Each entry has the following format.

```
/     =module_name    module_name
      =time_period    1
      =repetition     5
      =heavy          75
      =medium_heavy   60
      =medium         45
      =light          30
```

  You can modify this file to suit the needs of the module.

- a `load_control.tin` file that contains an entry for each module in your system—(In the fields, `rla` stands for "reduce load action" and `ila` stands for "increase load action.") Each entry has the following format.

```
/     =module_name          module_name
      =rla_threshold        heavy
      =rla_command_line     batch_admin  stop   normal
      =ila_threshold        medium
      =ila_command_line     batch_admin  start  normal
```

  You can modify this file to suit the needs of the module. You can also create additional `.tin` entries for a module.

The installation software also includes the command `load_control_admin login` in the `module_startup.cm` command file for each module.

Changing a Load Control Table

To change either load control table, perform the normal steps for updating a table file and installing the updated file. The example here follows the steps shown in the manual *OpenVOS System Administration: Configuring a System* (R287).

To have OpenVOS on a module recognize an updated load control table immediately rather than at the next bootload, specify the `reconfigure` request of the `load_control_admin` command. This command performs an immediate installation of both load control tables. To specify the `reconfigure` request for another module, specify the module name in the `-overseer_module` argument.

NOTE

You must be a privileged user to invoke the `load_control_admin` command.

For a description of two methods of giving this command on multiple modules, see the manual *OpenVOS System Administration: Configuring a System* (R287).

To modify and install the `load_control.tin` file, perform the following steps.

1. Issue the following command.

   ```
   change_current_dir (master_disk)>system>configuration
   ```

2. Edit the `load_control.tin` file and change one of the threshold values.
3. Issue the following commands.

   ```
   create_table load_control
   broadcast_file load_control.table >system
   load_control_admin reconfigure
   ```

After you perform the preceding steps, OpenVOS will recognize the updated table during the current bootload.

Enabling the Load Control Facility

To enable the load control facility, issue the following command.

```
load_control_admin login
```

The `-overseer_module` *`overseer_module`* argument of the `load_control_admin login` command lets you start a load control process on any module in the system. If you do not specify this argument, a load control process is logged in on the current module.

The `-login_priority` *`number`* argument of the `load_control_admin` command specifies a priority for the load control process that you are logging in. The priority should be the same as the priority of the average user of the module. If you do not specify this argument, the process will have priority 0.

The `-log` argument tells OpenVOS to write load control information into the file `load_control_log.(date)` in the directory `(master_disk)>system`. Load-control log file abbreviations are defined in Table 5-1.

Table 5-1. Load-Control Log File Abbreviations

Abbreviation	Description
`lf`	Load factor
`plf`	Previous load factor
`alf`	Average load factor
`tlf`	Total load factor

When the load control facility takes action to either increase or decrease the load, the `lf` figure in the log file becomes the `plf`.

The following command logs in a load control process with a priority of 6 on module `m8`.

```
load_control_admin login -login_priority 6 -overseer_module
m8
```

The following command logs in a load control process with the default priority of 0 on the current module, and tells OpenVOS to write information about the process to `load_control_log.(date)`.

```
load_control_admin login -log
```

Disabling the Load Control Facility

To disable the load control facility, issue the following command.

```
load_control_admin logout
```

The `-overseer_module` argument of the `load_control_admin logout` command lets you stop load control on any module in the system. If you do not specify this argument, the load control process is logged out on the current module.

The following command logs out the load control process on module `m3`.

```
load_control_admin logout -overseer_module m3
```

Creating a Load Control Histogram

The `load_control_histogram` command creates a histogram from the information in the load control log for a particular date. You can use the histogram to help determine if the threshold values in your load control action table are appropriate.

By default, the `load_control_histogram` command displays the histogram in terms of decisive load factors (that is, averages of the preliminary load factors). If you specify the `-no_avg_load` argument, OpenVOS displays the preliminary load factors.

Each line in the histogram contains the number of asterisks that corresponds to 68% of either the decisive or preliminary load factor. The example in the following section shows an excerpt from a load control histogram.

You can use the commands `start_logging` and `stop_logging`, documented in the *OpenVOS Commands Reference Manual* (R098), to log the histogram to a file.

Sample Load Control Histogram

The following command creates a histogram from the information in the load control log file for the current date. The histogram shows the preliminary load factors.

```
load_control_histogram -no_avg_load
```

```
00:01:34********
00:04:42*******
00:07:46******
00:10:49****
00:13:53****
00:16:58****
00:20:02***
00:23:06***
00:26:11**
00:29:15**
00:32:19**
00:35:24*
00:38:28*
00:41:34
00:44:37
00:47:41*
```

Chapter 6
The Remote Service Network

This chapter describes the Remote Service Network (RSN), which is a communications link between remote systems (that is, customer systems) and a module called the *HUB*. The HUB is the central point in the network, and all remote systems are connected to it. In most cases, the HUB is located in the Stratus Customer Assistance Center (CAC) or your authorized Stratus service representative.

This chapter contains the following sections:

You must make entries in the `devices.tin` file for the RSN connection. See *OpenVOS System Administration: Configuring a System* (R287) for information on these entries.

RSN Overview

In the RSN, processes distributed throughout a remote system communicate through message and server queues with a central process within that system. The central process then communicates with the HUB over an asynchronous dialup phone line or an Internet connection.

When the system is booted, the RSN is automatically started up by the `start_rsn` command in each module's `mddmon_start_up.cm` file (which, in turn, is started by the `start_stcp.cm` file). The various RSN-related processes then continue to run for the entire current bootload. (For information on the `mddmon_start_up.cm` file, see the manual *Migrating VOS Applications from Continuum Systems* (R607).)

The primary advantage of the RSN is that it automatically and immediately notifies the HUB of hardware failures in remote modules. In addition, the RSN provides the following services:

- Remote problem analysis.
- Accurate and up-to-date configuration management.
- Error, performance, and availability reports sent to the HUB.
- Electronic mail to and from the HUB.
- Efficient file transfers.
- Prompt reporting of software, hardware, and documentation problems

The RSN Bridge Module and the MaintBridgeServer Process

Each remote system has an RSN *bridge module* at the site. The bridge module contains a console server, which is typically configured at installation. The console server is one of the following types:

- RSN Internet console server, which has two Ethernet ports and serial ports. One Ethernet port typically connects to the bridge module. The other Ethernet port connects to a customer-provided router or a network that provides Internet connectivity. The serial port 1 connects to the COM1 port on the RSN bridge module. If you replace this console server, issue the `update_rsnip_site` command to configure it.
- RSN dialup console server, which has two serial ports and an Ethernet port. You typically connect one serial port to the COM1 port on the RSN bridge module and the Ethernet port to the bridge module. You connect the second serial port to the RSN modem, which connects to the dial-up public switched telephone network through a customer-provided telephone line. If you replace this console server, issue the `config_console_server` command to configure it.

The connection to bridge module from the Ethernet port (for either a dialup or Internet connection) is as follows:

- If the configuration includes a maintenance network Ethernet switch, the Ethernet port on the console server connects to it.
- If the configuration does not include a maintenance network Ethernet switch, but does include a network I/O enclosure (NIO), the Ethernet port on the console server connects to it.
- If the configuration has neither a maintenance network Ethernet switch nor an NIO, the Ethernet port on the console server connects directly to a PCI Ethernet port on the bridge module using a cross-over cable.

For complete information on the physical configuration of either the dialup console server or Internet console server, as well as information on replacing a console server, see the site planning guide as well as the operation and maintenance guide for your system, as listed in Related Manuals in the Preface.

The following sections provide additional information on the RSN bridge module:

- "The MaintBridgeServer Process" on page 6-3
- "Entries in STCP Database Files and in the `devices.tin` File" on page 6-4
- "Additional Information for the RSN Internet Console Server" on page 6-4
- "Changing from a Dialup Console Server to an Internet Console Server" on page 6-7

The MaintBridgeServer Process

The bridge module contains a server process called the MaintBridgeServer, the software connection between the HUB and the other modules in the remote system. The MaintBridgeServer process performs the following tasks.

- Listens for requests coming from the HUB.
- Notifies a module in the remote system when the HUB requires information from that module.
- Listens to modules in the system for information they want to transmit to the HUB.
- Calls the HUB when necessary.

If the MaintBridgeServer process cannot connect to the HUB on its first attempt, it tries at 15-minute intervals until it makes a connection (or until an administrator at your site discards the pending messages).

To create the MaintBridgeServer process, issue the `start_rsn` command in the `mddmon_start_up.cm` file on the bridge module. Note that if the module is a single-module system it is **always** the RSN bridge module.

The `monitor_rsn_servers` command checks if the MaintBridgeServer is running. This command is typically issued within a `start_process` command line in the `mddmon_start_up.cm` file

Entries in STCP Database Files and in the `devices.tin` File

The RSN bridge module requires the following entries in the `devices.tin` file.

- an entry for a incoming slave device (named `rsn_in.m`*X*, by convention), which handles incoming RSN calls from the CAC
- an entry for a outgoing slave device (named `rsn_out.m`*X*, by convention), which handles outgoing RSN calls to the CAC
- an entry for the RSN configuration device (named `rsn_cfg.m`*X*, by convention), which is used to manage the site ID and password.
- an entry for a connection to the COM1 port (named `rsn_terminal.m`*X*, by convention), which provides an alternative RSN connection.

The RSN bridge module also requires entries in two STCP database files (`services` and `telnetservice`) as well as the `mddmon_start_up.cm` file.

For detailed information on these entries, see the manual *OpenVOS System Administration: Configuring a System* (R287).

Additional Information for the RSN Internet Console Server

In an RSN configuration with an RSN Internet console server, the RSN bridge module requires an additional file and information, as the following sections describe:

- "Null-modem Configuration File" on page 6-4
- "Site ID and Password" on page 6-5
- "Setting the Timezone on the RSN Internet Console Server" on page 6-6

For information on security changes required in an RSN configuration with an RSN Internet console server, see "Firewall and SSL Concerns For the Internet Console Server" on page 6-11.

Null-modem Configuration File

The RSN Internet console-server configuration includes a null-modem configuration file, `null.modem`, which is located in the `(master_disk)>system` directory on the bridge module. You should never change this file. The pathname of the null-modem file, `(master_disk)>system>null.modem`, is the value for the `-dialer_type` argument of the `start_rsn` command that you issue to enable the RSN on a module. The value also appears in the `SITE PARAMETERS` screen of the `maint_request` command's `update` request.

Site ID and Password

For a module to communicate with the HUB, the module requires a unique site ID and password. You must issue the `update_rsnip_site` command to assign the site ID and password before starting or restarting the RSN bridge module.

NOTE

Before you issue the `update_rsnip_site` command, you **must** coordinate any changes in the value of the *`site_id`* argument and the value of the *`site_id`* argument of the `start_rsn` command with the CAC or HUB because the *`site_id`* values of these two commands and the *`site_id`* value stored at the HUB must all be identical.

A password is required for a connection between the HUB and the RSN Internet console server. The `update_rsnip_site` command uses the value specified by the *`rsn_connect_password`* argument to automatically update the value of the `RSN CONNECT PASSWORD` field on the `SITE PARAMETERS` screen.

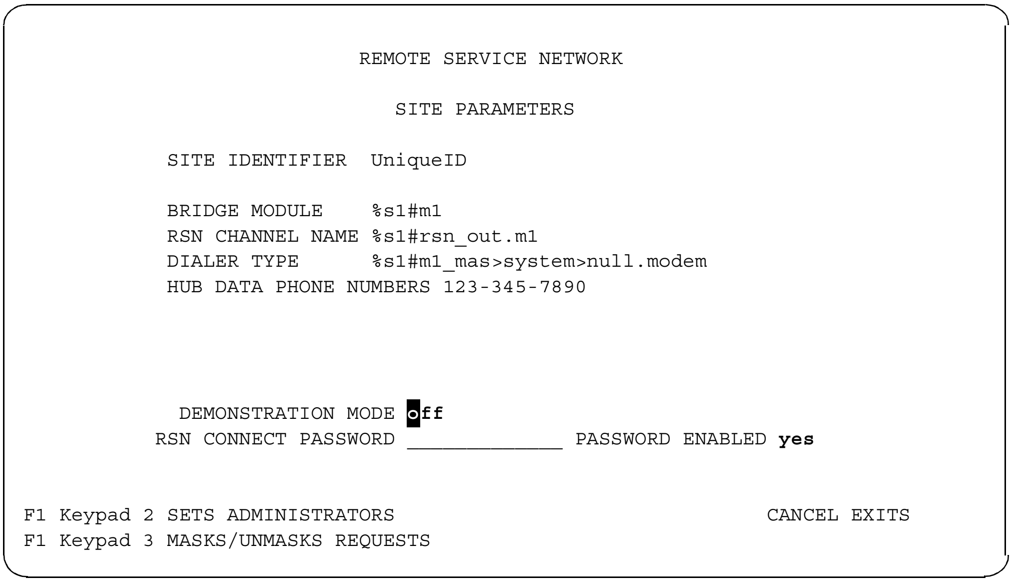
NOTES

1. You **must** coordinate any changes in the value of the `RSN CONNECT PASSWORD` field in the `SITE PARAMETERS` screen and, thus, the value of the *`rsn_connect_password`* argument with the CAC or HUB because these two values must be identical. For more information on this password, see "`rsn_connect_password` and `RSN CONNECT PASSWORD`" on page 14-191.

2. The value of the *`rsn_connect_password`* argument (and, thus, of the `RSN CONNECT PASSWORD` field) **is different** from the password of the RSN login name that HUB personnel use for interactive login sessions to the site. The RSN login name and password are stored in the site's `registration_admin` database and at the HUB, so you must also coordinate this information with the HUB.

To immediately update the site password, you can use the `SITE PARAMETERS` screen of the `maint_request` command's `update` request.

The `SITE PARAMETERS` screen for an RSN Internet console-server configuration is as follows:



```
                          REMOTE SERVICE NETWORK

                             SITE PARAMETERS

             SITE IDENTIFIER  UniqueID

             BRIDGE MODULE    %s1#m1
             RSN CHANNEL NAME %s1#rsn_out.m1
             DIALER TYPE      %s1#m1_mas>system>null.modem
             HUB DATA PHONE NUMBERS 123-345-7890




              DEMONSTRATION MODE off
            RSN CONNECT PASSWORD _____________ PASSWORD ENABLED yes


 F1 Keypad 2 SETS ADMINISTRATORS                              CANCEL EXITS
 F1 Keypad 3 MASKS/UNMASKS REQUESTS
```

If the `PASSWORD ENABLED` field is set to `no`, the `maint_request` command fails and displays the following message:

```
Could not update password on console server: RSN/IP requires
a password
```

The RSN Internet console server, the HUB, and OpenVOS **must** specify identical values for the site's RSN login (site ID) and password.

Setting the Timezone on the RSN Internet Console Server

You should set the timezone on the RSN Internet console server to be identical to the timezone of the OpenVOS module. To do so, perform the following steps:

1. On the RSN Internet console server, rename the existing `/etc/localtime` file to `UTC-localtime`.
2. Locate the correct time zone file on a computer running the Linux® operating system. The timezone file is a binary file that is typically located in the `/usr/share/zoneinfo` directory.

3. Copy the time zone file to the `/etc` directory on the Internet console server and name the file `*ZONE*-localtime`, where `*ZONE*` is the time zone of the OpenVOS module.
4. Using the Linux `ln` command in the `/etc` directory on the Internet console server, create a symbolic link `localtime` that points to the `*ZONE*-localtime` file that you created in step 3.
5. Issue the Linux `date` command on the Internet console server, to confirm that the new time zone appears.

Changing from a Dialup Console Server to an Internet Console Server

To change the RSN configuration from using a dial-up console server to an Internet console server, use the following procedure. This procedure assumes that the Internet console server is already customized for Stratus and that the configuration uses the network `10.10.1.0/24`. For detailed information on the physical placement of the console server, see the site planning guide. For detailed information on the cable connections, see the operation and maintenance guide for your system. The manuals are listed in Related Manuals in the Preface.

1. On an OpenVOS module, stop the RSN processes using the `stop_process` command or the `maint_request` command's `stop_server` request.
2. Disconnect the Ethernet cable from the dial-up console server and connect it to the Ethernet port designated for the local network on the Internet console server.
3. Connect another Ethernet cable to the Ethernet port designated for the wide-area network on the Internet console server, and then connect the other end of the cable to the public network.
4. Connect an RS-232 cable to serial port 1 (using the RJ-45 connector) on the Internet console server and to the COM1 port (using the DB-9 connector) on the OpenVOS module. Ports 2 through 8 on the console server should be disabled. Other than the connection to the public network, do **not** connect non-Stratus equipment to the console server or to the local maintenance network.
5. If the RSN Internet console server uses a name server, specify values for it in its `resolv.conf` file. To do so, contact the CAC.
6. Issue the `update_rsnip_site` command, which updates setup parameters on the Internet console server.
7. Modify the `start_rsn.cm` file by specifying the pathname of the null-modem file, `(master_disk)>system>null.modem`, as the value for the `-dialer_type` argument.
8. Start the RSN by issuing the `start_rsn` command.

The MaintServer Process

Every module in the remote system (including the bridge module) has a server process called the MaintServer. This process performs the following tasks.

- Watches for problems on the module.
- Listens for requests coming from the `maint_request` command.
- Notifies the MaintBridgeServer process whenever communication with the HUB is required.

To create this process, issue the `start_rsn` command in each module's `mddmon_start_up.cm` file.

The `monitor_rsn_servers` command checks if the MaintServer is running. This command is typically issued within a `start_process` command line in the `mddmon_start_up.cm` file

The `maint_request` Command

The `maint_request` command is the user interface to the RSN. This command has several requests that allow a local system administrator to control how the RSN functions at the site.

To use any of the `maint_request` command requests (except `mail`, `add_call`, and `update_call`), a user must be defined as a system administrator by an entry in the `HUB.data` file (described in "RSN Files and the `rsn_dir` Directory" on page 6-13). This file initially has one entry, the user name `*.SysAdmin`. A system administrator can modify the file by specifying the `update` request of the `maint_request` command.

The `maint_request` command and its requests allow you to perform the following tasks.

- Specify the `update` request to define other system administrators.
- Specify the `update` request to define which system administrators, if any, are notified about RSN interactions (mail and files received from the HUB). Each specified system administrator is notified on the 25th line of his or her terminal.
- Specify the `mask`, `unmask`, and `update` requests to control which types of requests your site accepts from the HUB. A request your site does not accept is called a *masked request*.
- Specify the `update` request to display a list of the masked requests.
- Specify the `update` request to define a password that the HUB must supply whenever it connects your site.

NOTE

You must coordinate any changes in the `RSN CONNECT PASSWORD` field in the `SITE PARAMETERS` screen of the `maint_request` command's `update` request with the CAC. For more information, see the "`rsn_connect_password` and `RSN CONNECT PASSWORD`" on page 14-191.

- Specify the `stop_server` request to stop communications between your site and the HUB.
- Specify the `truncate_queue` request to discard messages waiting to be sent to the HUB.
- Specify the `status` request to verify that the MaintBridgeServer process is running.
- Specify the `update` request to control the RSN's automatic service connections to the HUB when an error at your site is reported. This allows your site to perform demonstrations and tests to simulate failures.

The `mail`, `add_call`, and `update_call` requests of the `maint_request` command are deprecated but still supported. Instead of using these functions, use the appropriate requests in the Site Call System.

The Site Call System

The Site Call System provides an easy-to-use subsystem environment for accessing a call database at your site. It is a powerful tool for tracking calls and communicating with the HUB, and enhances the `add_call`, `update_call`, and `mail` requests.

The Site Call System database consists of the two files `site_call_db` and `site_call_text`. These files are located on the master disk of the bridge module in the directory `(master_disk)>system>site_call_system`. You must have modify access to this directory and write access to the `site_call_db` and `site_call_text` files.

Using the Site Call System, you can perform the following tasks:

- Add calls to the database.
- Update calls in the database.
- Research calls in order to solve problems without HUB involvement.
- Send and receive calls between your site and the HUB.

The Site Call System enables you to add application-specific issues to the database. These calls are assigned a tracking number, but remain local to your site. Should you require assistance, you may *escalate* the call at your discretion, which transmits the call to the HUB.

Also, all calls that are automatically generated by the RSN are added to your Site Call System database in addition to being sent to the HUB.

Updates made at the HUB to calls generated by your site are transmitted to your Site Call System database. This allows you to track the status of each call, review the on-going handling of each call, and provide timely call updates to help guide the handling of the call at the HUB.

To enter the subsystem, issue the `site_call_system` command. For detailed information about how to use the Site Call System, see the description of this command in Chapter 14.

Site Security

The RSN provides security measures so that your site administrator can control the flow of information between the site and the HUB, and restrict access to the RSN at the site. The following sections describe these security measures:

- "Masked Requests" on page 6-10
- "Disabling the RSN" on page 6-11
- "Passwords" on page 6-11
- "Users Who Are Not Administrators" on page 6-11
- "Sites That Do Not Allow the HUB to Dial In" on page 6-11
- "Three-Digit Validation Code For Voice Calls" on page 6-11
- "Firewall and SSL Concerns For the Internet Console Server" on page 6-11

Masked Requests

Your site administrator can prevent the HUB process from performing certain functions by placing, on a list of masked requests, the names of the requests that perform those functions. Your site ignores all masked requests received from the HUB.

To mask a request, your site administrator can issue the `mask` or `update` request of the `maint_request` command. To unmask a request, an administrator can issue the `unmask` or `update` request. To display a list of masked requests, an administrator can issue the `update` request.

The HUB functions that can be masked are explained in the `maint_request` command description in Chapter 14.

Disabling the RSN

Your site administrator can entirely disable the RSN at the site by issuing either the `stop_process` command or the `stop_server` request of the `maint_request` command.

If your site administrator disables the RSN, it is automatically enabled at the next bootload. An administrator can enable it during the current bootload with the `start_rsn` command.

Passwords

Your site administrator can specify a password that the HUB must supply whenever it connects to the site. The `RSN CONNECT PASSWORD` is always required to validate that the site is connecting to the correct HUB or that the HUB is connecting to the correct site. Note that an RSN Internet console server requires a password. For more information on this password, see "`rsn_connect_password` and `RSN CONNECT PASSWORD`" on page 14-191.

Users Who Are Not Administrators

You can create users who are not defined as administrators. These users can issue only the `mail`, `add_call`, and `update_call` requests of the `maint_request` command. Your site administrator defines other administrators at the site with the `update` request to the `maint_request` command.

Sites That Do Not Allow the HUB to Dial In

A site administrator who prevents the HUB from dialing into his/her site can specify the `periodic_call` request of the `maint_request` command to ensure the integrity of the communications connection. Alternatively, the site administrator can ask the HUB to turn on the heartbeat capability.

Three-Digit Validation Code For Voice Calls

A site administrator can request that a voice call coming into his/her site provide a three-digit code to verify that the caller originates from the HUB. The output of the `validate_hub` command displays this code for each individual site for a particular date. See Chapter 14 for information about the `validate_hub` command.

Firewall and SSL Concerns For the Internet Console Server

In an RSN configuration with an Internet console server, information is encrypted and is transferred between the Stratus call-home server and your site using the secure Sockets Layer (SSL) protocol. You must ensure that RSN traffic can reach the designated Stratus call-home server. In addition, your site needs to access the URL for the Certificate Authority in order to verify the SSL certificate and to obtain the latest certificate revocation lists.

To configure a successful RSN connection, you must modify the security on any firewall on your network to allow access to the sites listed in Table 6-1.

NOTES

1. Use the numeric IP addresses only if the firewall configuration requires a numeric address.
2. The IP addresses in Table 6-1 can change. To confirm the IP addresses, contact the Stratus CAC or your authorized Stratus service representatives.
3. For information about the designated Certificate Authority, contact the Stratus CAC or your authorized Stratus service representatives.

Table 6-1. URLs for the Internet Console Server

Site	URL/Host Name	IP Address	Port
Stratus Call-home Server	https://rsntunnel.ecacsupport.com	usual: 134.111.1.62 backup: 207.138.233.72	TCP 443
Certificate Authority	http://crl.netsolssl.com	205.234.175.175	TCP 80

For additional security and if your system is configured with OpenSSL and OpenSSH, you can use OpenSSH for the RSN connection from the bridge module to the Internet console server. However, use OpenSSH only if you are familiar with it and with the Linux operating system, and you require all data to be encrypted on your network. To use OpenSSH for the RSN connection, perform the following steps:

1. Generate an SSH public/private key on the OpenVOS bridge module:
   a. Create a user on the bridge module. (This user name is the value for the `-ssh_uid` argument of the `update_rsnip_site` command.) The purpose of this user is to provide SSH access to the bridge module from the Internet console server.
   b. Log in to the bridge module as this user.
   c. Create the directory `.ssh` as a subdirectory of the home directory.
   d. Change to the `.ssh` directory and then generate an SSH public/private key.

NOTE

You need to generate only RSA or DSA keys, but not both.

2. On the Internet console server, create the directory `/root/.ssh`.
3. Copy the generated keys to the Internet console server.
4. On the bridge module, delete the private key and rename the public key to `authorized_keys`.
5. Generate the `known_hosts` file by connecting to OpenVOS from the Internet console server, using the OpenSSH username and the IP address of the bridge module.
6. On the Internet console server, set file access rights to all the files in the directory `/root/.ssh` to `600` by issuing the command `chmod 600 *`.
7. When you issue the `update_rsnip_site` command, specify a value for the `-ssh_uid` argument.

If you use OpenSSH, the `telnetd` daemon must still run on the bridge module, though you can configure the `telnetd` daemon to listen only on the IP address of the bridge module, but do so, however, only if no other connections are using the TELNET protocol.

For information on configuring OpenSSL and OpenSSH on OpenVOS, see the *Software Release Bulletin: Internet Security Pack for OpenVOS, Release 4.0.2* (R660).

RSN Files and the `rsn_dir` Directory

The RSN facility includes several files and a directory that are used in RSN communication. These objects are automatically created by the RSN software and are stored in the `(master_disk)>system` directory.

The following files are required on every module.

- `HUB.config`

  This file keeps track of the module's configuration. With this information, the RSN can inform the HUB if a component fails while the module is shut down.

- `HUB.disk_data`

  This file contains data pertinent to the recent history of various disk errors such as `crc`, `record_not_found`, and total errors. The data stored in this file is used to diagnose disks that have exceeded a reasonable error threshold and should be replaced.

- `HUB.msg_SQ`

  This queue file is used by the `maint_request` command to communicate with the MaintServer process.

- `rsn_disk_threshold.table`

  This file holds the disk error threshold parameters. Do **not** try to use or modify it; it is intended for use only by the CAC.

The following files are required only on the bridge module.

- `HUB.comm_SQ`

  This file provides communication between the MaintServer and MaintBridgeServer processes.

- `HUB.comm_MQ`

  This file holds messages to be forwarded to the HUB, including those that are deferred or that cannot be forwarded immediately. To discard the messages in this file, issue the `maint_request` command's `truncate_queue` request.

- `HUB.data`

  This file contains control information about the current state of the RSN at the site. The data in the file includes a list of masked requests and a list of the user names that are defined as system administrators.

- `remote_maint_log.`*`date`*

  OpenVOS writes an entry to this file whenever either the HUB contacts the site, or the site contacts the HUB. You can use a switch in the `start_rsn` command to disable this facility for the current bootload, so that OpenVOS does not add any entries to the file.

  NOTE

  OpenVOS always sends notification of all RSN interactions to those administrators designated by the `update` request to receive notifications. This occurs even if the remote maintenance log is disabled for the bootload.

- `rsn_bridge_server.table`

  This file controls how the RSN is configured.

- `null.modem`

  This file is located in the `(master_disk)>system` directory on the bridge module. When the RSN configuration uses an Internet console server, the pathname of this file is the value for the `-dialer_type` argument of the `start_rsn` command that you issue to enable the RSN on a module. You should never change the `null.modem` file.

- `moxa.modem`

  This file is located in the `(master_disk)>system` directory on the bridge module. When the RSN configuration uses an Internet dialup console server (that is, a console server attached to a modem), the pathname of this file is the value for the `-dialer_type` argument of the `start_rsn` command that you issue to enable the RSN on a module.

In addition to the preceding files, the RSN facility includes a directory named `rsn_dir`, which holds files (such as mail) received from the HUB. The directory `rsn_dir` is automatically created on the bridge module the first time the RSN is brought up. It is stored in the `(master_disk)>system` directory of the bridge module.

Updating the RSN

Chapter 10 describes how to change the name of a module or system. If the module whose name you are changing is also the RSN bridge module, perform the following additional steps.

1. Edit the `start_rsn` line in the `mddmon_start_up.cm` file. Change the name of the bridge module so it reflects either the renamed system name or the renamed module name.
2. Specify the `update` request of the `maint_request` command to display the `HUB.data` file. Make a written note of the information it contains.
3. Delete the following files from the `(master_disk)>system` directory of the bridge module.

   ```
   HUB.config
   HUB.data
   rsn_bridge_server.table
   ```

4. Issue the `shutdown` command with the `-reboot` argument. During bootload, the `start_rsn` command automatically re-creates all RSN files, including `HUB.config`, `HUB.data`, and `rsn_bridge_server.table`.
5. Issue the `maint_request` command with the `update` request. Re-enter in the `HUB.data` file all of the information you recorded prior to rebooting.

The Mail Facility

You can send mail to, and receive mail from, the HUB from any module at a site by issuing either the `mail` request of the `maint_request` command or the `rsn_mail` command. Each letter sent from the HUB is packaged into a file named HUB_mail.date.time, where date and time are the current date and time.

The directory rsn_dir holds all files, including mail, that are sent from the HUB. Whenever you log in, check rsn_dir for mail from the HUB (or put a list command into your start_up.cm file to list the contents of rsn_dir). You can also list your mail using the rsn_mail command, which has the advantage of marking unread letters.

Specify the update request of the maint_request command to specify which system administrators are to be notified whenever a file or piece of mail is sent from the HUB to your site.

If you need a high level of security at your site, you can prevent the HUB from sending mail to your site by making the HUB's mail request a masked request, as explained in "Site Security."

The mail Request of the maint_request Command

The mail request of the maint_request command allows you to send mail to the HUB. If you do not specify a path name with the request, the command displays a form into which you can enter a short letter (up to 10 lines). If you specify a path name, the file specified is sent as mail to the HUB. Note that although you can send long files, it is wise to send only files shorter than about 4000 characters through the mail facility in order to keep the RSN communications line free for other activity.

NOTE

Stratus recommends that you use the rsn_mail request in the Site Call System.

The rsn_mail Command

The rsn_mail command enables you to easily send mail to and receive mail from the RSN. Letters can be read, printed, deleted, or sent to the HUB. This command provides all of the mail handling features of maint_request, plus some new functions. Using this command rather than the mail request of the maint_request command lets you keep track of which letters have been read. Any letter that has not been read is marked with an asterisk when you list your mail using the rsn_mail command.

The rsn_mail command, which accepts no arguments, displays a menu from which you can issue multiple requests. See the rsn_mail command description in Chapter 14 for more information.

Chapter 7
Updating Program Modules

OpenVOS includes two commands that enable you to track revisions of existing program modules. These commands, `create_pm_var_file` and `update_program_module`, enable you to modify program modules online without using tapes or transmitting entire program modules. Stratus personnel use these commands to compare program modules and send updated modules via the Remote Service Network (RSN) to remote sites.

This chapter describes how to compare two program modules and create a *variation file* that contains the differences. It contains the following sections.

The `create_pm_var_file` Command

The `create_pm_var_file` command compares two program modules and generates a variation file containing the differences. Depending on the extent to which the program modules differ, the variation file generally will be four to five times smaller than either of the compared program modules. Thus, if the largest of the compared program modules is 100 blocks long, the variation file may be only 25 blocks long. This reduction in size is very useful when distributing program modules over wide area networks, including the RSN.

The size of the variation file is a function of the number and size of differences between the two program modules being compared. Two program modules that vary greatly may produce a variation file very close in size to that of the larger of the original program modules. When the variation file exceeds 85% of the size of the requested program module, program execution stops and the system displays the following message.

```
Variation file has exceeded 85 percent of the upgrade
file size. Program execution halted.
```

The length of time it takes the command to execute generally depends on the number of differences found between the two program modules. When two files are similar, the files are compared quickly.

The variation file generated is named *`original_file.var`*. For example, if you want to compare `merge_directories1.pm` against `merge_directories2.pm`, the name of the variation file generated is `merge_directories1.var`.

The variation file is created and saved in the same directory in which the `create_pm_var_file` command is executed. The `create_pm_var_file` command will **not** process a program module larger than 4000 blocks.

For a complete description of the `create_pm_var_file` command, see Chapter 14.

The `update_program_module` Command

The `create_pm_var_file` command is used in conjunction with the `update_program_module` command to produce revisions of existing program modules. The `update_program_module` command reads the records in the variation file, created by the `create_pm_var_file` command, and creates the revised program module.

Before the command processes the difference records, it reads the variation file's header record to confirm it is updating the correct file. The program module created by the `update_program_module` command is saved in the same directory in which the command is executed. You can use the `update_program_module` command with program modules generated using VOS Release 7.0 or later.

For a complete description of the `update_program_module` command, see Chapter 14.

Updating Program Modules Online

The following steps demonstrate how to update a program module online. In these steps, two versions of the program `dump_database.pm` are compared. Note that the sample commands are intended for illustration purposes only; your path names and program module names will differ.

1. Issue the following `create_pm_var_file` command to compare the two `dump_database.pm` program modules and generate a variation file called `dump_database.var`.

```
create_pm_var_file
%s#m1>installed>commands>pm>dump_database.pm
%s#m1>development>commands>pm>dump_database.pm -statistics
```

The system displays the following output.

```
Loading program modules...

File A: %s#m1>installed>commands>pm>dump_database.pm
  Size: 8 blocks

File B: %s#m1>development>commands>pm>dump_database.pm
  Size: 9 blocks

Number of diffs: 278    Words of difference: 6333
Number of difference bytes: 14424 (4 blocks)

Percent size reduction: 61

Difference record distribution statistics:

    Size     Adds     Dels     Eqls     Mods
       1        1        3       59        1
       2        4        6       23       11
       5        3        1       42       33
      10        6        2       45       28
      50        0        0        0        5
     100        0        1        0        2
     500        0        0        0        0
    1000        2        0        0        0
    5000        0        0        0        0
    ****        0        0        0        0


               16       13      169       80

Clock Time: 5     CPU Time: 1.560104     Page Faults: 37
```

2. Transmit the variation file `dump_database.var` to a remote module, using a wide area network such as the RSN.
3. Stop any processes that are using the old program module, using the `stop_process` command.
4. At the remote module, issue the following `update_program_module` command to create the `newdump_database.pm` file by updating the `dump_database.pm` file with the `dump_database.var` variation file.

   `update_program_module`
   `%s#m13>installed>commands>pm>dump_database.pm`
   `%s#m13>installed>commands>pm>newdump_database.pm`

The system displays the following output.

```
Loading the original program module...

Original File: %s#m13>installed>commands>pm>dump_database.pm
         Size: 8 blocks

Clock Time: 1     CPU Time: 0.485291     Page Faults: 31
```

5. Install the updated program module.
6. Restart any processes that you stopped in step 3.

For more information, see the descriptions of the `create_pm_var_file` and `update_program_module` commands in Chapter 14.

Chapter 8
The Accounting Facility

This chapter describes the accounting facility, which writes accounting records to a file. *Accounting records* provide statistics about the resources used by the processes in a specified module. You can process the accounting data using a package that includes OpenVOS subroutines and a command for each OpenVOS language.

This chapter contains the following sections.

The `accounting_admin` Command

The accounting facility is enabled when you specify the `accounting_admin` command. The arguments specified in an `accounting_admin` command take effect as processes are created; therefore, it is best to issue the command in your `module_start_up.cm` file. To disable accounting again during a login session, specify the `-disable_accounting` argument of the `accounting_admin` command.

When you first enable the facility on a module, OpenVOS creates a directory named `accounting` in the `(master_disk)>system` directory on that module's master disk.

Accounting Logs

A file into which OpenVOS writes accounting records is an *accounting log*. OpenVOS creates a new accounting log each day (beginning at midnight) that accounting is enabled on a module. Accounting logs are named `raw_acct_log.(date)`, where `(date)` is the current date. They are stored in the `accounting` directory. The Overseer creates the accounting log as an extended sequential file (ESF), which enables the file to grow to a size of approximately 128 GB.

Once every hour, on or near the turn of the hour, the Overseer process verifies the date and time. If this verification occurs during the hour before midnight, the Overseer process creates a new accounting log that includes the next day's date. Any records for the remaining part of the current day are added to the new log. The records are date- and time-stamped so that the accounting facility can search the appropriate logs for each day's accounting records.

Accounting Records

When accounting is enabled on a module, accounting records are always written at the following times.

- at process creation
- at process termination
- when OpenVOS cleans up a terminated process
- when the parameters of the accounting facility are altered

In addition, you can select arguments with the `accounting_admin` command that tell OpenVOS to write accounting records at other times, such as when a process closes a file or executes a command. You can also include in the accounting records statistics about the I/O activity on ports.

Every raw accounting record contains the following information.

- the record type
- the process ID
- the time the record was written to the log
- a sequence number, so that missing records can be detected

Types of Accounting Records

There are seven types of accounting records, each of which is described in the following sections.

- "Default Process Logging" on page 8-3
- "Command Logging" on page 8-3
- "Extended Process Logging" on page 8-3
- "File Logging" on page 8-4
- "Transaction Logging" on page 8-5
- "Port Logging" on page 8-5
- "Administration Logging" on page 8-6

Default Process Logging

Default process logging creates records for each process running on a specified module. To enable default process logging, issue the `accounting_admin` command with the `-disable_accounting` argument either set to `no` (the default) in the display form or omitted in the command-line form.

Table 8-1 summarizes default process logging.

Table 8-1. Default Process Logging

Event to Be Logged	Time Event Logged
Process start	When a process is created
Process termination	When a process is terminated
Process cleanup	When OpenVOS cleans up a terminated process
Process log	When records are written for processes that exist in more than one accounting log

Command Logging

Command logging collects records for each command running on a specified module. To enable command logging, issue the `accounting_admin` command with the `-log_commands` argument.

Command logging can write a large number of records to the accounting log in a short time. When command logging is enabled, you must monitor disk usage, as described in the "Monitoring Disk Usage" section later in this chapter.

Table 8-2 summarizes command logging.

Table 8-2. Command Logging

Event to Be Logged	Time Event Logged
Command start	When a command begins execution
Command termination	When a command terminates execution

Extended Process Logging

Extended process logging creates additional records for each process running on a specified module. Extended process logging creates the following accounting records.

- process-statistics records—These provide additional information about processes. For example, these records show how much time a server process spends performing functions for each process it serves.

- process-defined records—These are records whose content is defined by the user; they allow the user to insert data into an accounting log.

You must write a program to process these records.

To enable extended process logging, issue the `accounting_admin` command with the `-log_proc_stats_records` and `-log_proc_user_records` arguments either set to `yes` (the default) in the display form or omitted in the command-line form.

Table 8-3 summarizes extended process logging.

Table 8-3. Extended Process Logging

Event to Be Logged	Time Event Logged	`accounting_admin` Command Argument
Process statistics	When a process calls `s$log_resource_usage`	`-log_proc_stats_records`
Process defined	When a process calls `s$log_process_record`	`-log_proc_user_records`

File Logging

File logging creates records for the open files on a specified module. The accounting facility creates file-close records whenever file logging is enabled. More detailed file-log records, however, are created only when either or both of the following types of logging are also enabled.

- *command logging*, which you enable by specifying the `-log_commands` argument
- *extended process logging*, which you enable by specifying the `-log_proc_stats_records` argument

To enable file logging, issue the `accounting_admin` command with the `-log_files` argument.

NOTE

File logging can write a large number of records to the accounting log in a short time. When file logging is enabled, you must monitor disk usage, as described in the "Monitoring Disk Usage" section later in this chapter.

Table 8-4 summarizes file logging.

Table 8-4. File Logging

Event to Be Logged	Time Event Logged	`accounting_admin` Command Argument
File close	When a file is closed (either implicitly or explicitly)	`-log_files`
File log (commands)	Before command-start and command-termination records are written	`-log_files` and `-log_commands`
File log (process)	Before process-termination, process-statistics, and transaction records are written	`-log_files` and `-log_proc_stats_records`

Transaction Logging

Transaction logging creates records for each transaction running on a specified module. To enable transaction logging, issue the `accounting_admin` command with the `-log_transactions` argument.

When transaction logging is enabled and the module is running a transaction-processing application that is heavily used, you must monitor disk usage, as described in the ‘‘Monitoring Disk Usage” section later in this chapter.

Table 8-5 summarizes transaction logging.

Table 8-5. Transaction Logging

Event to Be Logged	Time Event Logged
Transaction start	When a transaction is initiated
Transaction commit	When a transaction is committed
Transaction termination	When a transaction is terminated

Port Logging

Port logging creates records for each I/O port on a specified module. Specifically, port logging records the number of bytes of data being read and written by each I/O port, as well as the total number of reads and writes per process by each I/O port.

To enable port logging, issue the `accounting_admin` command with the `-port_accounting` argument.

Table 8-6 summarizes port logging.

Table 8-6. Port Logging

Event to Be Logged	Time Event Logged
I/O port read	When an I/O port reads data
I/O port write	When an I/O port writes data

Administration Logging

Administration logging creates a record describing the parameters of the current `accounting_admin` command. Administration logging is enabled by default when you issue the `accounting_admin` command. Subsequent changes to the parameters of the `accounting_admin` command are automatically recorded in the administration record.

Monitoring Disk Usage

File logging, command logging, and transaction logging can each cause a large number of records to be written to the accounting log in a short period of time. If you are running the accounting facility with any of these types of logging enabled, you must monitor disk usage so that the disk containing the accounting log does not fill up. For this purpose, use the commands `display_disk_info` and `display_disk_usage`, both described in the *OpenVOS Commands Reference Manual* (R098).

The `dump_accounting_file` Command

The `dump_accounting_file` command is provided within the accounting facility as a debugging tool. It displays on the screen or writes to a file some or all of the records in an accounting log.

Processing the Accounting Logs

OpenVOS provides a subroutine package you can use to read accounting logs and produce reports. The package reads selected records from the specified logs and presents them to the caller in chronological order. While reading the records, it prepares a database; other subroutines in the package can then retrieve the summary information.

OpenVOS also provides the include files necessary to support the package, and sample programs in OpenVOS PL/I and OpenVOS Pascal that use the subroutine package.

Chapter 9
Customizing Status Codes and Messages

OpenVOS enables you to customize your module's status codes and status messages so that they provide information specific to your operating environment. This chapter describes how to customize your module's status codes and status messages. It contains the following sections.

Status Code Definitions

There are four types of status codes.

- error codes, whose names begin with `e$`
- message codes, whose names begin with `m$`
- query codes, whose names begin with `q$`
- response codes, whose names begin with `r$`

Each status code is associated with a name, number, and text that identifies the meaning of the code and whether it is a predefined code or a user-defined code. The codes are stored in files that are shipped with the installation software.

The Status Code Files

Five files deal with status codes. The following four files must be stored in the directory `(master_disk)>system>error`.

- `error_codes.dd`

  A predefined file that you cannot change. It contains the format for the records that are in the `error_codes.table` file. This file is part of the installation software.

- `user_codes.tin`

  Specifies the contents of user-defined records in the `error_codes.table` file. Each record description follows the format of the `error_codes.dd` file. You modify this file when you define your own status codes. The `user_codes.tin` file is **not** part of the installation software; you must create it.

- `error_codes.tin`

  Specifies the contents of those records in the `error_codes.table` file that are defined by OpenVOS. Each record description follows the format of the `error_codes.dd` file. You modify this file to change the text of an OpenVOS status message. This file is part of the installation software. Keep a printed copy of `error_codes.tin` in case you ever have to look up the message text corresponding to the status code numbers and you cannot access them online.

- `error_codes.table`

  A table file that contains records from both `error_codes.tin` and `user_codes.tin`.

  The `error_codes.table` file is **not** part of the installation software; you must create it by issuing the `update_codes` command. Chapter 14 describes the `update_codes` command.

The fifth file, the `error_codes.text` file, is contained in the `(master_disk)>system` directory. It is a compressed version of the `error_codes.table` file, containing only the text of the status messages. The `error_codes.text` file is included in the installation software. It is updated by the `update_codes` command.

The `error_codes.dd` File

The `error_codes.dd` file, which you must **never** modify, appears as follows:

```
fields:    num       binary (15),
           name      char (32),
           text      char (128) varying,
           category  char (18),
           category2 char (18);
end;
```

This file defines the record format for both the `error_codes.tin` and `user_codes.tin` files.

Defining Additional Status Codes

The numbers between `16000` and `18000` have been reserved for you to define status codes for your system in addition to those provided by OpenVOS. To define your own status codes, you must perform several steps, including creating and editing a `user_codes.tin` file. The following sections describe the file and explain the steps.

- "The Fields in the `user_codes.tin` File" on page 9-3
- "Sample `user_codes.tin` File Entry" on page 9-4
- "Defining or Modifying Your Own Status Codes" on page 9-4

The Fields in the `user_codes.tin` File

The fields in the `user_codes.tin` file are as follows:

▶ `num`

Specifies the number of the status code. The value must be in the range from `16000` to `18000`. The first code assigned must be `16000`.

▶ `name`

Specifies the name of the status code. The name can only contain the characters `$`, `_`, `0-9`, `A-Z`, and `a-z`. The name must have a prefix of `e$`, `m$`, `q$`, or `r$`. The prefix conventions show the context in which a status code is used; `e$` specifies an error code returned by a subroutine, `m$` specifies a message written by a command, `q$` specifies a question, and `r$` specifies the valid responses to a question.

- `text`

  Specifies a character string specifying the message that the status code displays. The string must be enclosed in apostrophes. The text of a `q$`, `m$`, or `r$` status message can contain parameters of the form `& a`*`n`*`&` (where *`n`* is an integer from `1` to `3`) to be substituted for actual values on output. An `e$` error message cannot contain replaceable arguments. An `e$` message must end with a period (`.`) or an exclamation point (`!`). A `q$` query message must end with a question mark (`?`), a question mark followed by a space (`? `), a colon (`:`), or a colon followed by a space (`: `).

- `category`

  Specifies the type of the status code. The value must be `user_error_codes`.

- `category2`

  Not relevant to users. Provides for internal reclassification of status codes.

Sample `user_codes.tin` File Entry

An entry in a `user_codes.tin` file might appear as follows:

```
/ =num           16000
  =name          e$msg_unattachable
  =text          'This message cannot be placed in this queue.'
  =category      user_error_codes
```

Defining or Modifying Your Own Status Codes

To define or modify your own status codes, perform the following steps.

1. Change your current directory to the `(master_disk)>system>error` directory.
2. Edit the `user_codes.tin` file. (If the file does not already exist, you can create it using a text editor.) Using the format shown in the preceding section, add or modify one or more entries, then save the file.
3. Issue the command `update_codes`.
4. Install the file `error_codes.text` in the `(master_disk)>system` directory of every module in the system. The modified error file will be available at the next bootload.

To successfully execute the `update_codes` and `broadcast_file` commands, you must have modify access to the `(master_disk)>system` and `(master_disk)>system>error` directories, and you must have default write access to the files in those directories.

You can test the new message file in your own process with the `use_message_file` command, described in the *OpenVOS Commands Reference Manual* (R098).

NOTE

If you have created your own status codes, you must invoke the `update_codes` command during every major and update installation of OpenVOS to incorporate them into the `error_codes.text` file. Invoke the command **after** the pre-boot phase of the installation and **before** you shut down and reboot the module.

The following example illustrates how to add the user-defined error code `e$msg_unattachable` to the `error_codes.text` file.

1. Issue the `change_current_dir (master_disk)>system>error` command.
2. Edit the `user_codes.tin` file. Add an entry for error code `16000`, `e$msg_unattachable`, and save the file.
3. Issue the `update_codes` command.
4. Issue the `broadcast_file error_codes.text >system` command.

Modifying the Text of a Status Message

To modify the text of an OpenVOS status message, follow the steps listed in the previous section, "Defining or Modifying Your Own Status Codes." Instead of editing the `user_codes.tin` file, edit the `error_codes.tin` file.

The following sections describe the fields in the `error_codes.tin` file and explain how to change the text of an OpenVOS status message.

- "The Fields in the `error_codes.tin` File" on page 9-5
- "Sample `error_codes.tin` File Entry" on page 9-6

The Fields in the `error_codes.tin` File

The fields in the `error_codes.tin` file are as follows:

▶ `num`

Specifies the status code itself, assigned by OpenVOS. You **cannot** modify this field.

▶ `name`

Specifies the name of the status code. You **cannot** modify this field.

- `text`
  Specifies a character string specifying the message that the status code displays. The string must be enclosed in apostrophes. This is the only field that you can modify. For a description of how to format the text field, see the section "The Fields in the `user_codes.tin` File" earlier in this chapter.

- `category`
  Specifies the type of the status code. You **cannot** modify this field.

- `category2`
  Not relevant to users. Provides for internal reclassification of status codes.

Sample `error_codes.tin` File Entry

An entry in `error_codes.tin` appears as follows:

```
/     =num          1002
      =name         e$bad_disk
      =text         'Undefined disk name.'
      =category     os_error_codes
```

The following example illustrates how to change the text of the preceding error message.

1. Issue the `change_current_dir (master_disk)>system>error` command.
2. Edit the `error_codes.tin` file. For error code `1002`, change the `text` field from `'Undefined disk name.'` to `'Nombre del disco indefinido.'`
3. Issue the `update_codes` command.
4. Issue the `broadcast_file error_codes.text >system` command.

Setting System Defaults in the `error_codes.tin` File

When you modify the text field of a status code, you are modifying the text displayed by a status code. However, modifying this field also changes the system default values of the `-mapping_rules` argument of compiler commands.

The values for the `-mapping_rules` argument are stored in the following `error_codes.tin` entries.

```
/    =num         4992
     =name        m$shortmap
     =text        'shortmap'
     =category    lang_message_codes

/    =num         4993
     =name        m$shortmap_check
     =text        'shortmap/check'
     =category    lang_message_codes

/    =num         4994
     =name        m$longmap
     =text        'longmap'
     =category    lang_message_codes

/    =num         4995
     =name        m$longmap_check
     =text        'longmap/check'
     =category    lang_message_codes
```

You must modify the following `error_codes.tin` entry in order to reset the default value of the `-mapping_rules` argument.

```
/    =num         4991
     =name        m$default_mapping
     =text        'shortmap'
     =category    lang_message_codes
```

For more information on the `-mapping_rules` argument in compiler commands, see the *OpenVOS Commands Reference Manual* (R098).

Chapter 10

Modifying Module and System Identifiers

This chapter describes how to modify important module and system identifiers. Specifically, it explains how to modify the module or system name, the module, station, or system number, and the site ID.

This chapter contains the following sections.

Overview

The Open StrataLINK product provides cross-module communications via the Streams TCP/IP product (STCP). ftServer and Continuum-series multimodule systems require Open StrataLINK. Multisystem Open StrataLINK also supports cross-system communications among all module types, along with StrataNET. If you are running any of these cross-module and cross-system communications products, you must update your configuration files in some circumstances.

Before modifying module and system identifiers, be aware of the following restrictions that may affect cross-module and cross-system communications products running on your system:

- The `modules.tin`, `systems.tin` and `backbone_systems.tin` files are obsolete. Use the `new_modules.tin`, `new_systems.tin`, and `new_backbone_systems.tin` files instead.
- If your applications use hardcoded names, renaming modules and systems may result in problems.
- Renaming or renumbering modules and systems will disrupt the network operation, so avoid making such changes unless absolutely necessary.

For information about Open StrataLINK, see the *VOS Administrator's Guide for Open StrataLINK* (R388). For information about StrataNET, see the manual *VOS Communications Software: X.25 and StrataNET Administration* (R091). For information about STCP, see the *OpenVOS STREAMS TCP/IP Administrator's Guide* (R419).

Renaming a Module or a Master Disk

This section describes how to modify the name of a module or a master disk. For a description of the location of the master disk, see ''The Master Disk'' on page 1-1.

NOTE

While you are performing the following procedure, other users who are logged in to the system may encounter unexpected error messages. Therefore, to avoid confusion, it is a good practice to have users log off the system **before** you start performing the procedure, even though logging off is not required until Step 13.

To modify the name of a module, perform the following steps.

1. Update the label of the boot disk, using the `update_disk_label` command. This step is necessary because a module's descriptive information is derived from the label of its boot disk rather than from the `new_modules.table` file.

   NOTES

   1. The boot disk must be a RAID disk; it cannot be a JBOD disk.
   2. When you specify the `update_disk_label` command, you may receive error messages similar to the following. You can safely ignore these messages.

      ```
      update_disk_label: Invalid system
      number. When checking value of
      sys_info$time_zone on target
      system/module.
      ```

   When using the `update_disk_label` command, specify `(master_disk)` for the *`disk_name`* argument and specify the new module name for the `-module_name` argument. If you plan to modify a master disk name, use the `-new_disk_name` argument. The command then writes the new name(s) to the label of the disk named in the command.

The `update_disk_label` command is documented in the manual *OpenVOS System Administration: Disk and Tape Administration* (R284).

2. Update the modules table with the new name of the module.

   With ftServer modules, you must use Open StrataLINK and the new table file `new_modules.table`.

   a. Modify the module's entry in the `new_modules.tin` file on the master module. If you have a single-module system, this module is the master module.
   b. Re-create the `new_modules.table` file with the `create_table` command, and install it on each module in the system by broadcasting it to the `(master_disk)>system` directory with the `broadcast_file` command.

   You need not issue the `configure_modules` command on the master module, since you shut down and reboot the module in Step 13, and the new disk label information takes effect then.

3. Update the disks and devices tables with the module's new name.

   a. Update the `disks.tin` and `devices.tin` files on the master module to reflect the new module name. If you have a single-module system, this module is the master module.
   b. Re-create the `disks.table` and `devices.table` files with the `create_table` command, and install them on each module in the system by broadcasting them to the `(master_disk)>system` directory with the `broadcast_file` command.

   You do not need to issue the `configure_disks`, or `configure_devices` commands on the master module, since you shut down and reboot the module in Step 13, and the new disk label information takes effect then.

4. If your system has a gateway to a wide area network, update the gateways table with the module's new name. If your system does not have an X.25 gateway, go to Step 5.

   a. Update the `gateways.tin` file on the master module to reflect the new module name. If you have a single-module system, this module is the master module.
   b. Re-create the `gateways.table` file with the `create_table` command, and install it on each module in the system by broadcasting it to the `(master_disk)>system` directory with the `broadcast_file` command.

   You need not issue the `configure_gateways` command on the master module, since you shut down and reboot the module in Step 13.

5. If, in renaming your module, you are **not** modifying the name of its master disk, skip this step and go to Step 6.

   If you **are** renaming the master disk, invoke the `translate_links` command, specifying the current master disk name in the `-root_dir` argument. This command modifies the disk name throughout the directory hierarchy.

   Note that a disk name has the following two components.

   - the name of the system, prefixed by a percent sign (`%`)
   - the name of the disk, prefixed by a number sign (`#`)

   For example, a disk name might be `%s1#d01`.

   The `translate_links` command is documented in the *OpenVOS Commands Reference Manual* (R098).

   NOTICE

   Do not try to rename the master disk during a release installation. The installation software will try to update software libraries using the old disk name.

6. Update the `maintsw_config.tin` file with the new system and module names, and then create a new `maintsw_config.table` file with the `create_table` command. For details, see *Migrating VOS Applications from Continuum Systems* (R607).
7. Update the `start_stcp.cm` file with the new system and module names. For details, see the *OpenVOS STREAMS TCP/IP Administrator's Guide* (R419).
8. If the module you are renaming uses STCP TELNET services, update the `telnetservice` file with the new devices. For details, see the *OpenVOS STREAMS TCP/IP Administrator's Guide* (R419).
9. Rename the `installation_log.db` file in the `(master_disk)>system>release_dir` directory of a single-module system or, in the case of a multimodule system, in the `(master_disk)>system>release_dir` directory on every module in the system.

   If you do not rename this file, and you are also renaming the master disk, you will encounter problems when you reboot the module. OpenVOS obtains library path names from the `installation_log.db` file rather than calculating new path names directly. But because those path names contain the old disk name, they are invalid, and OpenVOS will be unable to find the proper system libraries.

NOTICE

Do **not** delete this file; the CAC may need it later if you encounter problems. For more information on the `installation_log.db` file, see the *OpenVOS Installation Guide* (R386).

10. If either the module you are renaming or any module in the same system is running the Transaction Processing Facility (TPF), perform the following steps.
    a. Ensure that no TPF application is executing.
    b. Stop the TPOverseer process using the `stop_process` command. The `stop_process` command is documented in the *OpenVOS Commands Reference Manual* (R098).
    c. Specify the following commands:

    ```
    change_current_dir >system
    rename transaction_logs transaction_logs.(date).(time)
    ```

    You restart the TPOverseer process by issuing the `start_process` command in the `module_start_up.cm` file when you reboot the module in Step 13. See the *VOS Transaction Processing Facility Guide* (R215) for further information on TPF administration.

11. If the module whose name you are modifying serves as the bridge module to the Remote Service Network (RSN), modify RSN-related files, as described in "Updating the RSN" on page 6-15.

    NOTE

    If the module is a single-module system it is **always** the RSN bridge module.

12. If, in renaming your module, you are **not** renaming its master disk, skip this step and go to Step 13.

    If you **are** renaming the master disk, and it contains home directories, update the registration database to reflect the new name, as described in this step.

    Create a special session password with the `login_admin` command, then update the registration database to insert the new disk name into any home directory path names that contain the old disk name.

    Creating the special session password is necessary to prevent other users on your system from logging in between the time you update the registration database and the time the module name modification becomes effective (following the system's shutdown and reboot procedure in Step 13). Any user allowed to log in would experience difficulty because OpenVOS would be unable to locate the redefined home directory.

Users already logged in when you update the registration database are unaffected during their current login.

NOTE

This special session password is necessary any time you rename a disk that contains home directories.

The manual *OpenVOS System Administration: Registration and Security* (R283) describes how to modify the registration databases. This manual also documents the `login_admin` command.

13. If all users have not already logged off, warn any users remaining on your system that you are going to shut down and reboot the system, as described in the manual *OpenVOS System Administration: Starting Up and Shutting Down a Module or System* (R282). Then, on any module in your system, issue the following command.

Renaming a System

This section describes how to modify the name of a system.

To modify the name of a system, perform the following steps.

1. Update the label of the boot disk on each module in the system, using the `update_disk_label` command. This step is necessary because a system's descriptive information is derived from the labels of its modules' boot disks rather than from the `new_systems.table` file.

   NOTE

   The boot disk must be a RAID disk; it cannot be a JBOD disk.

   When using the `update_disk_label` command, specify the full path name of the logical volume to which the disk belongs in the format `%`*`system_name`*`#`*`disk_name`* (for example, `%sys#m1_mas`). Also, specify the new system name using the `-system_name` argument. The command then writes the new name to the label of the specified module's master disk.

   For a description of logical volumes and for more information about the `update_disk_label` command, see the manual *OpenVOS System Administration: Disk and Tape Administration* (R284).

2. Update the systems table with the system's new name.

   With ftServer modules, you must use Open StrataLINK and the `new_modules.table`, `new_systems.table`, and `new_backbone_systems.table` files.

   a. Modify the system's entry in the `new_systems.tin` file on the master module.
   b. Re-create the `new_systems.table` file with the `create_table` command, and install it on each module in the system by broadcasting it to the `(master_disk)>system` directory with the `broadcast_file` command.

   You need not specify the `configure_systems` command, since you shut down and reboot every module in the system in Step 14, and the new disk label information takes effect then.

3. If you are **not** renaming a system in a cluster network, skip Steps 3 and 4 and go directly to Step 5.

   If you **are** renaming a system in a cluster network, update the backbone systems table with the name of the new system.

   a. Modify the system's entry in the `new_backbone_systems.tin` file on the master module.
   b. Re-create the `new_backbone_systems.table` file with the `create_table` command and install it on the local bridge module.

   You need not specify the `configure_systems` command, since you shut down and reboot every module in the cluster network in Step 14, and the new disk label information takes effect then.

4. Update the network access table with the name of the new system.

   a. Modify the system's entry in the `network_access.tin` file on the local bridge module.
   b. Re-create the `network_access.table` file with the `create_table` command and install it on the local bridge module.

   You need not specify a configuration command for the network access table; OpenVOS recognizes this table as soon as it is installed.

5. If you are **not** renaming a system in a multisystem StrataNET or Open StrataLINK network, skip Steps 5, 6, and 7, and go directly to Step 8.

   If you **are** renaming a system in a multisystem StrataNET or Open StrataLINK network, update the nodes table with the name of the new system.

   a. Modify the system's entry in the `nodes.tin` file on the master module.

b. Re-create the `nodes.table` file with the `create_table` command, and install it on each module in the system by broadcasting it to the `(master_disk)>system` directory with the `broadcast_file` command.

You need not specify the `configure_systems` command, since you shut down and reboot every module in the system in Step 14, and the new disk label information takes effect then.

6. Update the network access table with the name of the new system.

   a. Modify the system's entry in the `network_access.tin` file on the master module.

   b. Re-create the `network_access.table` file with the `create_table` command, and install it on each module in the system by broadcasting it to the `(master_disk)>system` directory with the `broadcast_file` command.

   You never need to specify a configuration command for the network access table; OpenVOS recognizes this table as soon as it is installed.

7. Re-create and install an updated network routes table.

   a. Issue the `build_routes` command on the master module and install the `network_routes` file on the master module.

   b. Install the new `network_routes` file on each module in the system by broadcasting it to the `(master_disk)>system` directory with the `broadcast_file` command.

   You need not specify the `configure_routes` command, since you shut down and reboot every module in the system in Step 14, and the new disk label information takes effect then.

8. Create a special session password with the `login_admin` command, then update the registration database to insert the new system name into any home directory path names that contain the old system name.

   Creating the special session password is necessary to prevent other users on your system from logging in between the time you update the registration database and the time the system name modification becomes effective (following the system's shutdown and reboot procedure in Step 14). Any user logging in would experience difficulty because OpenVOS would be unable to locate the redefined home directory.

   NOTE

   Users already logged in when you update the registration database are unaffected during their current login.

The manual *OpenVOS System Administration: Registration and Security* (R283) describes how to modify the registration databases. This manual also documents the `login_admin` command.

9. Issue the `translate_links` command once for each disk in the system, specifying the disk name in the `-root_dir` argument. This command modifies the disk name throughout the directory hierarchy.

   Note that a disk name has two components.

   - the name of the system, prefixed by a percent sign (`%`)
   - the name of the disk, prefixed by a number sign (`#`)

   For example, a disk name might be `%s1#d01`.

   The `translate_links` command is documented in the *OpenVOS Commands Reference Manual* (R098).

10. Rename the `rsn_bridge_server.table` file from the `(master_disk)>system` directory.

    If you do not rename this file, you will have problems when you reboot the modules.

11. Rename the `installation_log.db` file from the `(master_disk)>system>release_dir` directory on each module in the system.

    If you do not rename this file, you will have problems when you reboot the modules. OpenVOS obtains library path names from the `installation_log.db` file rather than calculating new path names directly. But because those path names contain the old system name, they are invalid, and OpenVOS will not find the proper system libraries.

    NOTICE

    Do **not** delete this file; the CAC may need it later if you encounter problems. For more information on the `installation_log.db` file, see the *OpenVOS Installation Guide* (R386).

12. Perform the following steps on every module that is running TPF.

    a. Ensure that no TPF application is executing.

    b. Stop the TPOverseer process using the `stop_process` command. The `stop_process` command is documented in the *OpenVOS Commands Reference Manual* (R098).

c. If desired, save the transaction logs that are found in the module's `(master_disk)>system>transaction_logs` directory with the transaction-protected files.

d. Delete the old transaction logs from the `transaction_logs` directory.

e. Remove all links in the `transaction_logs` directory by issuing the following command:

```
unlink *
```

You restart the TPOverseer process by issuing the `start_process` command in the `module_start_up.cm` file when you reboot the module in Step 14. See the *VOS Transaction Processing Facility Guide* (R215) for further information on TPF administration.

13. If the module whose name you are changing is also the RSN bridge module, perform the additional steps described in "Updating the RSN" on page 6-15.

14. Warn any users remaining on your system that you are going to shut down and reboot the system, as described in the manual *OpenVOS System Administration: Starting Up and Shutting Down a Module or System* (R282). Then, on any module in your system, issue the following command.

```
shutdown -module * -reboot
```

This command shuts down and reboots all modules in the system so that they read their boot disk labels and any new configuration tables.

The `shutdown` command is documented in the manual *OpenVOS System Administration: Starting Up and Shutting Down a Module or System* (R282).

15. If you are **not** renaming a system in a cluster network, skip this step and go directly to Step 16.

If you **are** renaming a system in a cluster network, modify the configuration tables of all other systems in the cluster network to reflect the system's new name. Modify the following tables.

- `network_access.table`
- `new_backbone_systems.table`
- `new_systems.table`

Shut down and reboot these systems so that the name modification takes effect. See Step 14 for instructions.

NOTICE

If you cannot reboot the modules on the other systems in your network, contact the CAC for assistance.

16. If you are **not** renaming a system in a multisystem StrataNET or Open StrataLINK network, skip this step.

    If you **are** renaming a system in a multisystem StrataNET or Open StrataLINK network, modify the configuration tables of all other systems in the network to reflect the new system name. This includes the following tables.

    - `network_access.table`
    - `network_routes`
    - `new_systems.table`
    - `nodes.table`

    Shut down and reboot these systems so that the name modification takes effect. See Step 14 for instructions.

NOTICE

If you cannot reboot the modules on the other systems in your network, contact the CAC for assistance.

Modifying a Module and/or Station Number

This section describes how to modify a module and/or station number.

1. Update the label of the boot disk, using the `update_disk_label` command. This step is necessary because a module's descriptive information is derived from the label of its boot disk rather than from the `new_modules.table` file.

   NOTE

   The boot disk must be a RAID disk; it cannot be a JBOD disk.

   When using the `update_disk_label` command, specify `(master_disk)` for the *`disk_name`* argument and specify one or both of the following arguments.

   ```
   -module_no new_module_number
   -station_no new_station_number
   ```

The command then writes the new number(s) to the label of the disk named in the command.

The `update_disk_label` command is documented in the manual *OpenVOS System Administration: Disk and Tape Administration* (R284).

2. Update the modules table with the new module and/or station number.
   a. Modify the module's entry in the `new_modules.tin` file on the master module. If you have a single-module system, this module is the master module.
   b. Re-create the `new_modules.table` (or `modules.table`) file with the `create_table` command, and install it on each module in the system by broadcasting it to the `(master_disk)>system` directory with the `broadcast_file` command.

   You need not specify the `configure_modules` command on the master module, since you shut down and reboot the module in Step 6, and the new disk label information takes effect then.

3. If you are modifying **only** a station number, skip this step and go to Step 4.

   If you are modifying a module number, and that module or any module in the same system is running the TPF, perform the following steps.

   a. Ensure that no TPF application is executing.
   b. Stop the TPOverseer process using the `stop_process` command. The `stop_process` command is documented in the *OpenVOS Commands Reference Manual* (R098).
   c. If desired, save the transaction logs that are found in the module's `(master_disk)>system>transaction_logs` directory with the transaction-protected files.
   d. Delete the old transaction logs from the `transaction_logs` directory.
   e. Remove all links in the `transaction_logs` directory by issuing the following command.

      ```
      unlink *
      ```

   You restart the TPOverseer process by issuing the `start_process` command in the `module_start_up.cm` file when you reboot the module in Step 6. See the *VOS Transaction Processing Facility Guide* (R215) for further information on TPF administration.

4. If you are modifying **only** a module number, or the module does **not** serve as a bridge module in a cluster network or multisystem StrataNET or Open StrataLINK network, skip this step and go to Step 6.

If you are modifying a station number, and the affected module serves as a bridge module in a cluster network or multisystem StrataNET network, update the relevant systems table with the new station number.

a. Modify all occurrences of the station number in the `new_systems.tin` file on the master module in the system containing the affected module.
b. Re-create the `new_systems.table` file with the `create_table` command, and install it on each module in that system by broadcasting it to the `(master_disk)>system` directory with the `broadcast_file` command.

You need not specify the `configure_systems` command, since you shut down and reboot every module in the system in Step 6, and the new disk label information takes effect then.

5. Update the relevant backbone systems tables with the new station number.

   a. Modify all occurrences of the station number in the `new_backbone_systems.tin` file on the affected bridge module.
   b. Re-create the `new_backbone_systems.table` file with the `create_table` command and install it on the affected bridge module.
   c. Using the `copy_file` command, copy the updated `new_backbone_systems.tin` and `new_backbone_systems.table` files into the corresponding directories on each of the other bridge modules in the cluster network.

   You need not specify the `configure_systems` command, since you shut down and reboot the systems in the network in Step 7, and the new disk label information takes effect then.

6. Warn any users remaining on the system containing the affected module that you are going to shut down and reboot the system, as described in the manual *OpenVOS System Administration: Starting Up and Shutting Down a Module or System* (R282). Then, on any module in the system containing the affected module, issue the following command.

   ```
   shutdown -module * -reboot
   ```

   This command shuts down and reboots all modules in the system so that they read their boot disk label and/or the new configuration tables.

   The `shutdown` command is documented in the manual *OpenVOS System Administration: Starting Up and Shutting Down a Module or System* (R282).

7. Perform this step **only** if you are modifying a station number, and the affected module serves as a bridge module in either a cluster network or a multisystem StrataNET or Open StrataLINK network.

Shut down and reboot the systems in the network so that the number modification takes effect.

> NOTICE
>
> If you cannot reboot the modules on the other systems in your network, contact the CAC for assistance.

Renumbering a System

To modify a system number, perform the following steps.

1. Update the label of the boot disk on each module in the system, using the `update_disk_label` command. This step is necessary because a system's descriptive information is derived from the labels of its modules' boot disks rather than from the `new_systems.table` file.

   > NOTE
   >
   > The boot disk must be a RAID disk; it cannot be a JBOD disk.

   When using the `update_disk_label` command, specify `(master_disk` *`module_name)`* for the *`disk_name`* argument and specify the new system number for the `-system_no` argument. The command then writes the new number to the label of the specified module's master disk.

   The `update_disk_label` command is documented in the manual *OpenVOS System Administration: Disk and Tape Administration* (R284).

2. Update the systems table with the new system number.

   a. Modify the system's entry in the `new_systems.tin` file on the master module.

   b. Re-create the `new_systems.table` file with the `create_table` command, and install it on each module in the system by broadcasting it to the `(master_disk)>system` directory with the `broadcast_file` command.

   You need not issue the `configure_systems` command, since you shut down and reboot every module in the system in Step 5, and the new disk label information takes effect then.

3. If you are **not** renumbering a system in a cluster network, skip this step and go to Step 4.

   If you **are** renumbering a system in a cluster network, update the backbone systems table with the new system number.

   a. Modify the system's entry in the `new_backbone_systems.tin` file on the local bridge module.
   b. Re-create the `new_backbone_systems.table` file with the `create_table` command and install it on the local bridge module.

   You need not issue the `configure_systems` command, since you shut down and reboot every module in the cluster network in Step 5, and the new disk label information takes effect then.

4. Perform the following steps on every module that is running the TPF.
   a. Ensure that no TPF application is executing.
   b. Stop the TPOverseer process using the `stop_process` command. The `stop_process` command is documented in the *OpenVOS Commands Reference Manual* (R098).
   c. If desired, save the transaction logs that are found in the module's `(master_disk)>system>transaction_logs` directory with the transaction-protected files.
   d. Delete the old transaction logs from the `transaction_logs` directory.
   e. Remove all links in the `transaction_logs` directory by issuing the following command.

      ```
      unlink *
      ```

   You restart the TPOverseer process by issuing the `start_process` command in the `module_start_up.cm` file when you reboot the module in Step 5. See the *VOS Transaction Processing Facility Guide* (R215) for further information on TPF administration.

5. Warn any users remaining on your system that you are going to shut down and reboot the system, as described in the manual *OpenVOS System Administration: Starting Up and Shutting Down a Module or System* (R282). Then, on any module in your system, issue the following command.

   ```
   shutdown -module * -reboot
   ```

   This command shuts down and reboots all modules in the system so that they read their boot disk labels and any new configuration tables.

   The `shutdown` command is documented in the manual *OpenVOS System Administration: Starting Up and Shutting Down a Module or System* (R282).

6. If you are **not** renumbering a system in a cluster network, skip this step and go to Step 7.

If you **are** renumbering a system in a cluster network, modify the `new_systems.table` and `new_backbone_systems.table` files of all other systems in the cluster network to reflect the new system number.

Also shut down and reboot these systems so that the modification takes effect.

> NOTICE
>
> If you cannot reboot the modules on the other systems in your network, contact the CAC for assistance.

7. If you are **not** renumbering a system in a multisystem StrataNET or Open StrataLINK network, skip this step.

   If you **are** renumbering a system in a multisystem StrataNETor Open StrataLINK network, modify the `new_systems.table` file of all other systems in the network to reflect the new system number.

   Shut down and reboot these systems so that the modification takes effect.

   > NOTICE
   >
   > If you cannot reboot the modules on the other systems in your network, contact the CAC for assistance.

Modifying Your Site ID

Your site ID is a unique character string that identifies your site to various Stratus support facilities and serves as the key to the support services they provide.

> NOTICE
>
> Modifying your site ID without notifying Stratus may cause delays in service and problems using Stratus licensed software.

Before modifying your site ID, do one of the following:

- Call the CAC by phone and advise them of the modifications you want to make. They will open a call about your request and ask your local systems engineer to contact you. For a list of CAC phone numbers, see https://www.stratus.com/services-support/customer-support/.
- Call your local systems engineer directly.

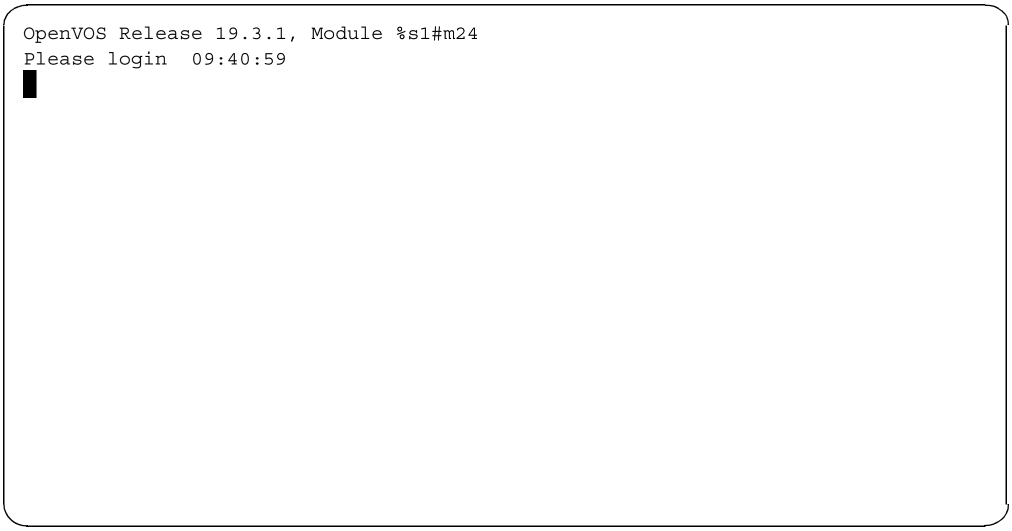
Your systems engineer will advise you on modifying the site ID.

To determine your current site ID, use one of the following methods.

- Locate the `start_rsn` command line that appears in the file `(master_disk)>system>mddmon_start_up.cm` file (your module start-up command macro). The first argument value specified on the command line after the command name is your site ID.
- Invoke the `validate_hub` command. The first line of output specifies the site ID.

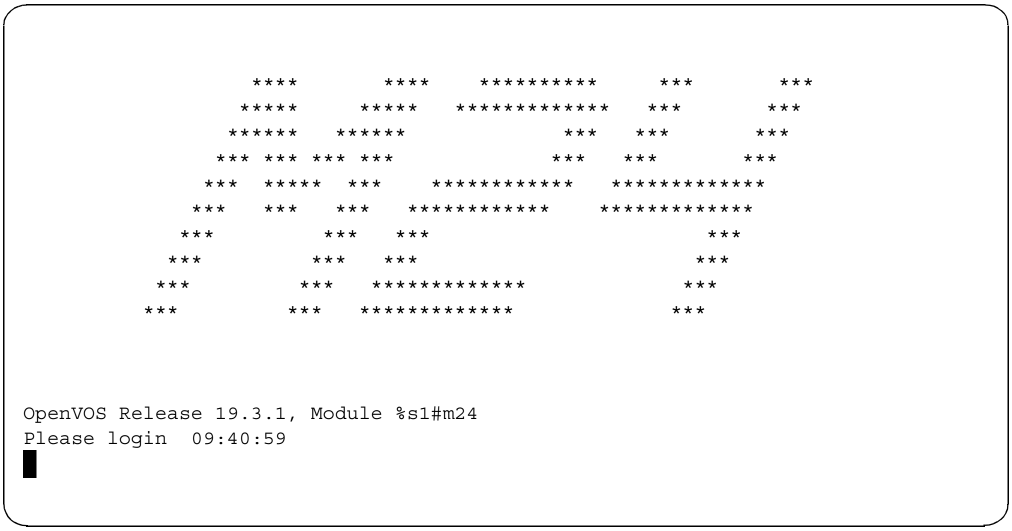
Chapter 11
Customizing Your Module's Prelogin Screen

A terminal that is turned on, but not logged into by a user, displays a *prelogin screen*. When OpenVOS is first installed, the prelogin screen displays only a login message, as illustrated in the following example.



```
OpenVOS Release 19.3.1, Module %s1#m24
Please login  09:40:59
█
```

You can customize your module so that its prelogin screen shows both a login banner and a login message, as illustrated in the following example.



```
                   ****       ****    **********     ***       ***
                  *****     *****   *************   ***       ***
                 ******   ******             ***   ***       ***
                *** *** *** ***             ***   ***       ***
               ***  *****  ***    ************   *************
              ***   ***   ***   ************    *************
             ***         ***   ***                       ***
            ***         ***   ***                       ***
           ***         ***   *************             ***
          ***         ***   *************             ***



OpenVOS Release 19.3.1, Module %s1#m24
Please login  09:40:59
█
```

The prelogin screen always contains the `Please login` message. The contents of the rest of the prelogin screen depend on whether you have placed a file named `login_screen_image` in the directory `(master_disk)>system` on your module. When OpenVOS displays a prelogin screen, it first looks for this file. If it exists, OpenVOS displays it and then displays the `Please login` message.

To produce a login banner, perform the following steps.

1. Create a file named `login_screen_image`.
2. Using a text editor, create the image you want to display. Note that the file should be no longer than 21 lines, because the combination of login banner and login message will not fit on your terminal’s screen.
3. Move the `login_screen_image` file into the directory `(master_disk)>system` on your module.

The contents of the file `login_screen_image` is now used as the login banner whenever OpenVOS displays a prelogin screen.

If you want to minimize the amount of prelogin information the system displays on the screen, you can edit the text of the `m$login_banner` code to remove the module name and OpenVOS release identifier from the login banner.

Note that placing a `login_screen_image` file in the system directory of **one** module in your system does not provide a login banner for **all** modules in your system. To display login banners on the other modules in your system, do one of the following:

- Create a separate `login_screen_image` file for each of the other modules in the system and place it in the corresponding module's `(master_disk)>system` directory. Using separate `login_screen_image` files allows you to create different login banners for each module in your system.
- Create links from the `(master_disk)>system` directories of the other modules in your system to an existing `login_screen_image` file. In this case, the login banners are the same for the modules using that file.

Chapter 12
Network Time Protocol

The Stratus implementation of the Network Time Protocol (NTP) is based on NTP Release 4.2.8p3. NTP is an Internet protocol used to synchronize the time of one computer with another computer or a reference time source (for example, modem clock, atomic clock, GPS clock) that provides Universal Time Coordinated (UTC).

This chapter contains the following sections:

This section describes OpenVOS-specific NTP information. For general information about NTP, see `www.ntp.org`.

NTP Licensing Information

Licensing information for NTP follows:

NTP Commands

The NTP installation process places the following commands in the `(master_disk)>system>command_library` directory:

- `ntpd.pm`—The NTP command that sets and maintains the system time of day in synchronicity with Internet standard time servers.

  NOTE

  If you specify `ntpd -N`, the priority level is not affected.

- `ntpdate.pm`—Sets the local date and time by polling the NTP server(s) specified as the `server` arguments to determine the correct time. This command's functionality is now available in the `ntpd` command.
- `ntpq.pm`—Queries the daemon about its state. You use this command to display information about the daemon.
- `ntpdc.pm`—Similar to `ntpq.pm`, but you can use it to modify the daemon's operating parameters without stopping and restarting the daemon.

NTP Configuration Information

Important configuration information follows:

- The OpenVOS installation process places the following configuration-related files in the `(master_disk)>system>ntp` directory. Note that *`release`* represents the current OpenVOS release number (for example, `sample_ntp.conf.17.1`):
  - `sample_ntp.conf.release`
  - `sample_start_ntpd.release`
- The file `(master_disk)>system>release_dir>sample_module_start_up.release` contains information about starting the daemon as part of booting a module.
- The TCP `services` file requires port 123 for UDP, both on the server and on the client.
- The NTP daemon requires root permission.
- The daemon requires a configuration file, typically named `ntp.conf`, that is typically located in `(master_disk)>system>ntp`. An example configuration file follows.

```
server serverone.xyzcompany.com
logfile >system>ntp>ntp.logfile
driftfile >system>ntp>ntp.drift
```

The preceding example configures the server named `serverone` as a time server (you specify one `server` directive for each time server). The example also tells the daemon to create a logfile named `ntp.logfile` in `(master_disk)>system>ntp` and to create a driftfile named `ntp.drift` in `(master_disk)>system>ntp`.

NTP Runtime Information

By default, NTP sets the time if local time differs from the server by more than 128 milliseconds. After NTP sets the time, it manages differences by *slewing* the time (that is, by varying the clock frequency). Because OpenVOS strongly discourages large time jumps, you should use the `ntpd -x` argument. This argument raises the step/slew boundary to 600 seconds (10 minutes), thus requiring you to make sure that your module is within 10 minutes of the correct time. If you do not specify the `-x` argument, a time delta that is greater than 128 milliseconds will set the time.

If OpenVOS is required to slew a large amount of time (that is, minutes), OpenVOS slews at a rate that NTP finds unacceptable, causing the daemon to periodically send messages about excessive frequency delta to the log. Such large adjustments may take days to converge to the server time. As the module time converges to within seconds of the server time, the messages stop. When the module and server times finally converge, a module reboot will be off by only one or two seconds.

After the module time converges, the daemon next adjusts the frequency delta (also called the *drift*). This value, expressed in parts per million (PPM), is an adjustment to the module time to keep it synchronized with the server time. The value is updated to the drift file every hour and used by a reboot to establish the initial frequency delta.

Time Synchronization Guidelines

Every OpenVOS module should use NTP to synchronize with an Internet time service. An NTP-based time service maintains time more accurately and does so across an entire network. Organizations typically provide a radio- or GPS-based clock that is protected by an Internet firewall and is Internet accessible to the OpenVOS module.

Before OpenVOS support of NTP, OpenVOS modules used the `set_jiffy_times` command to synchronize time across all modules in a system. If all OpenVOS modules in a system use NTP, `set_jiffy_times` is not needed, and in fact interferes with and degrades NTP time synchronization.

If any OpenVOS module in a system cannot use NTP, you must implement an alternate synchronization model. In the model, a single OpenVOS module uses NTP to maintain time synchronization and then periodically executes `set_jiffy_times` to synchronize the other modules. A typical update interval is once per day. This is the only legitimate use of `set_jiffy_times`.

Chapter 13
Tuning the Disk Cache

You might be able to improve performance by tuning the disk cache appropriately for your application. Not all applications require large caches, and some applications may benefit from having more memory than cache. For example, a messaging application may require more memory than cache for message buffers, while a large database application may require more cache.

When you tune the cache, you must maintain a balance between optimum performance of your system and ensuring that modified data in the cache is safely written to disk. You must also balance performance improvement (achieved when you increase the cache size, thereby reducing disk I/O) with the potential degradations that can occur if the system runs out of available memory. This chapter discusses issues surrounding these balancing acts and contains information on tuning the cache using the `set_tuning_parameters` command and the `set_cache_param` request of the `analyze_system` subsystem. This chapter also presents information about monitoring cache activity using `analyze_system` requests. For information on the `set_tuning_parameters` command, see the command description in Chapter 14. For information on requests of the `analyze_system` subsystem, see the *OpenVOS System Analysis Manual* (R073).

NOTE

In this chapter, *disk* refers to the logical unit number (LUN) of an ftScalable Storage system. An ftScalable Storage LUN is sometimes referred to as a *volume*.
ftScalable Storage LUNs (that is, volumes) appear as disks to OpenVOS. The maximum number of LUNs that OpenVOS can support is 254.

This chapter contains the following sections.

- "The Cache Manager" on page 13-2
- "Per-Volume Attributes" on page 13-8
- "Tuning the Cache" on page 13-11

- “Additional Tuning Issues” on page 13-28
- “RAM Files” on page 13-36

Administrators who tune the cache must be familiar with OpenVOS internals, patching variables, tuning issues, and I/O performance. Before making any changes, be sure to measure performance under typical and heavy load, and measure performance again after making the changes. Contact Stratus Professional Services if you need assistance.

The Cache Manager

The cache manager handles all aspects of the cache. The cache manager is an integral part of OpenVOS and interacts with other OpenVOS system components, including `file_io`, directory management, the transaction processing facility, and OpenVOS system commands such as `copy_file`. The goal of the cache manager is to provide quick and efficient access to a virtually unlimited number of disk blocks using a limited amount of memory.

There are two major aspects of cache performance:

1. Utilize memory to minimize the number of disk reads required.
2. Manage disk I/O to minimize the latency of any single disk operation.

To minimize disk reads, the cache manager uses a least recently used (LRU) algorithm. When the cache becomes full, the cache manager evicts the LRU blocks from the cache to make room for new blocks.

To minimize disk latency, the cache manager uses a combination of tuning parameters and patchable variables, as this chapter describes.

The cache manager provides the following features:

- Tuning parameters, modified through the `set_tuning_parameters` command. Changes to these parameters take effect immediately.
- Cache control on a system-wide or per-disk basis.
- Monitoring tools and meters that allow you to view, monitor, and analyze the workings of the cache manager using `analyze_system` requests.
- Ability to allow applications to communicate directly with the cache manager, to control cache on a per-file basis, taking file characteristics into account. This ability allows an application to designate its file blocks with an infinite unreferenced grace time (*memory-resident files*) or cause its file blocks to pass quickly through the cache (*transient files*), thus avoiding eviction of other blocks that are more likely to be candidates for reuse.
- Ability to utilize process priority to affect cache policy.

- Retention of expired blocks, up to the minimum cache size.
- Support of very large and very small cache sizes.
- Ability to designate preread policy on a file-by-file basis.
- Ability to handle non-persistent RAM files utilizing all available physical memory without incurring disk I/O.

The cache manager provides effective cache-tuning methods for optimizing application performance. The following sections discuss general strategies for tuning the cache using the `set_tuning_parameters` command and `analyze_system` requests.

- "Memory Available to the Cache Manager" on page 13-3
- "Grace Times" on page 13-6
- "Cache-Manager Timer" on page 13-6
- "Process Delays" on page 13-7

Memory Available to the Cache Manager

The cache manager can access up to approximately 10 GB of memory per module. (10 GB is a physical limitation due to the current implementation and is not related to available physical memory.) However, to avoid a negative impact on performance, the cache manager is limited to using a total amount of memory, as follows:

- Virtual memory—The cache manager can consume up to half of the virtual memory that is available during the bootload. You cannot increase the virtual memory limit to a greater value, and you should not need to lower the virtual memory limit.
- Physical memory—The cache manager can consume up to 30% of the first 128 MB of physical memory in the module, plus up to 95% of the remaining physical memory. By default, OpenVOS uses 60% of the remaining physical memory. This default ensures that OpenVOS has sufficient memory available to respond to changing system demands.

Using the maximum of 30% of the first 128 MB and the default of 60% of the remaining physical memory, the cache manager has available approximately 142,000 blocks for each gigabyte of memory. Table 13-1 shows the results of this formula for various memory configurations, and the approximate number of cache blocks.

Table 13-1. Memory Available to Cache Manager

Memory Size (GB)	Default Number of Cache Blocks	Settable Number of Cache Blocks
2 GB	283,593	283,593
4 GB	419,430	576, 562
16 GB	419,430	2,742,680

The default cache size for large memory configurations is approximately 1.7 GB and is not proportionally higher when more memory is available. You need to use the `set_tuning_parameters` command to explicitly set the cache to a larger size.

Although you cannot change the percentage of the first 128 MB that OpenVOS uses, you can set the percentage applied to physical memory over 128 MB to determine the maximum physical memory limit to use for the cache. Use the `-cm_cache_mem_pct` argument of the `set_tuning_parameters` command to set the percentage. You can specify a percentage from 0 to 95.

In Table 13-1, the column **Settable Number of Cache Blocks** shows the maximum number of blocks that the cache can contain using the default of 60%. You can use substantially more cache memory by increasing this to 95%. However, regardless of memory available for cache, 3,145,728 is the maximum number of blocks that you can set for the cache and 3,145,728 is the maximum value that the `set_tuning_parameters` command will accept. The actual maximum value used is 2,490,368, due to a cache implementation limit.

The following diagram illustrates the physical limits of pages of memory and how you can tune them. (The diagram illustrates the relative location of the cache size parameters, but does not indicate the actual physical locations in memory.)

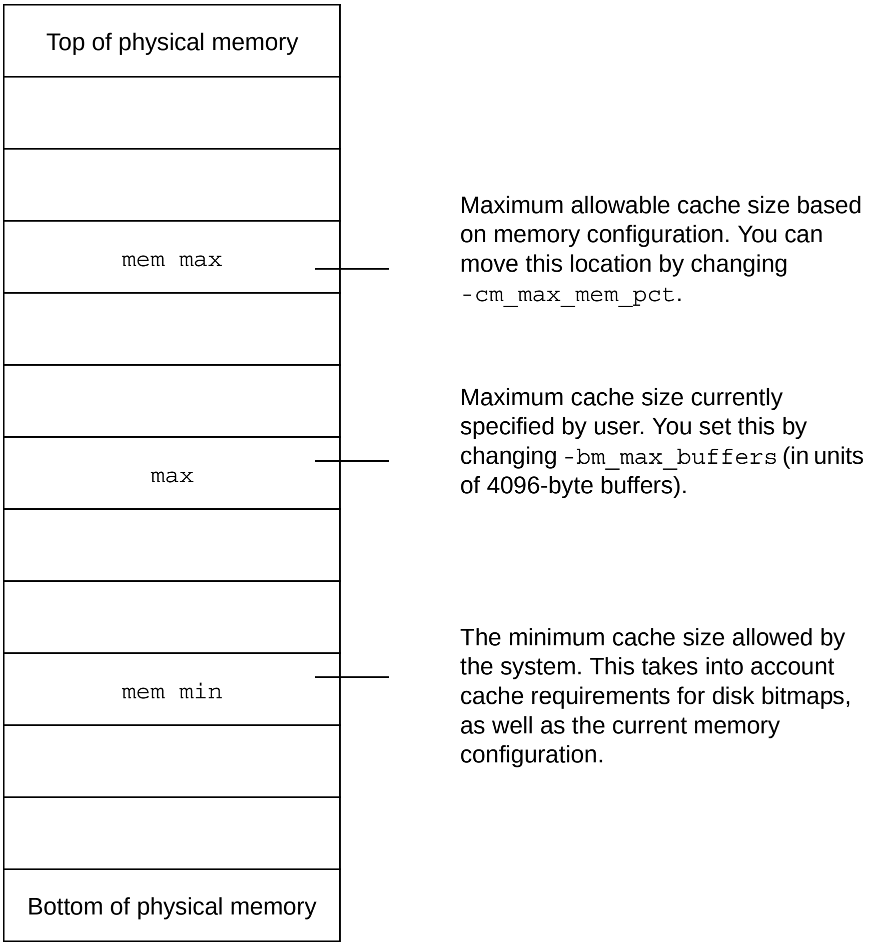

Top of physical memory
mem max
max
mem min
Bottom of physical memory
Maximum allowable cache size based on memory configuration. You can move this location by changing -cm_max_mem_pct.
Maximum cache size currently specified by user. You set this by changing -bm_max_buffers (in units of 4096-byte buffers).
The minimum cache size allowed by the system. This takes into account cache requirements for disk bitmaps, as well as the current memory configuration.

Figure 13-1. Pages of Memory

You can display these values using the `dump_cache_info` request of the `analyze_system` subsystem. This request also displays the actual values being used. Note that the minimum cache size can be less than the value of `mem min` shown in Figure 13-1, since the minimum cache size value represents a target value. The actual minimum is equal to the `mem min` in this case, but because conditions may change (for example, disks are removed, thereby reducing required bitmaps), the real minimum may likewise change.

This is also true for the maximum cache size which can be set to a higher value than `mem max`. The actual maximum cache size will be `mem max` or possibly lower if there is sufficient demand on physical memory.

Grace Times

Grace times enable you to make the cache manager more efficient for your site by factoring application-specific information into the formula that the cache manager uses to determine how frequently it writes to disk or frees unused buffers. The cache-manager formula operates in two different modes, depending on the current environment.

- When the system has sufficient memory resources to satisfy current demands, the grace times affect how the cache manager evicts cache blocks based on their state (as listed below) and the age of the data in the block. This algorithm enables the system to avoid unnecessary disk reads and writes.
- When the cache is in a fully committed state, the cache manager ignores the grace times. Instead, it implements the algorithm that finds the LRU block (that is, the LRU algorithm) to determine which blocks it will evict from the cache in order to satisfy new requests.

You can modify how the cache manager treats certain types of cache blocks in the formula by changing values for one or more of the grace-time arguments.

Throughout its life, a block of data goes through one or more of the following states:

- free
- referenced
- unreferenced
- modified
- unmodified

In general, the cache manager attempts to keep readable data in the cache as long as possible while the data is in use (referenced) and uses free or unreferenced blocks when it needs new blocks. The amount of time that the cache manager retains modified data in the cache is a balance between minimizing the number of times the system must write to disk when a block is rewritten and updating blocks on disk to reflect blocks changed in the cache.

Cache-Manager Timer

The cache manager uses a process called the cache-manager timer (`cmt`) to update the cache according to expiration rules, which you can modify using the `set_tuning_parameters` command. The `cmt` applies the expiration rules to phys. A *phy* (or *phys* for more than one data structure) is a data structure that the cache manager uses to track physical disk blocks. A phy can be in one of various states: unused, referenced, unreferenced, and modified. An unused or unreferenced phy is referred to as a *free* phy. The cache manager also uses a virtual address, *vir* (or *virs* for more than one virtual address), to pass data between system address space and the disk subsystem. More phys may be available than virs, but you can control the limits of phys and virs using the `set_tuning_parameters` command. These limits affect the expiration rules that the `cmt` uses to update the cache.

The notion of expiration is key to cache management, and phys that are in different states have different expiration criteria. For example, certain expiration criteria apply to modified phys, based on how long modified phys remain in cache before they are written to disk, and other expiration criteria apply to unreferenced phys, based on when the memory needed to track unreferenced phys is returned to the system for general use.

The `cmt` also uses other limits to update the cache. When a file is closed or when the system is shutdown, a very large number of modified blocks in the cache can cause long flush times while blocks are written to disk. Thus, a limited number of modified phys are allowed in cache on a per-disk basis. This limited number is referred to as the *modified block limit* or simply the *mod limit*. By default, the limit is determined for each disk, based on actual write performance. The limit is set so that all modified blocks can be written to that disk within approximately 30 seconds, a value that you can set using the `-max_flush_time` parameter of the `set_cache_param` request of the `analyze_system` subsystem. This is referred to as *dynamic modified block limitation*. If instead you want the cache manager to use a static limit, the `mod_limit` argument of the `set_cache_param` request allows you to change the default behavior so that a fixed limit is applied to all disks without regard for performance feedback. With static modified block limitation, the value defined by the `-cm_disk_mod_limit` argument of the `set_tuning_parameters` command defines this static limit. For dynamic modified block limitation, this value serves only as the starting point for subsequent adjustment.

The `cmt` runs intermittently to enforce modified block limits and other expiration criteria. Under normal circumstances, the `cmt` is scheduled to run every 5 seconds. When the `cmt` cannot enforce expiration criteria within 5 seconds, it is rescheduled to run every second. You can change these time intervals using parameters such as `min_cmt_sec` and `max_cmt_sec` of the `set_cache_param` request of the `analyze_system` subsystem.

Process Delays

If processes are modifying blocks at a faster rate than the disk queues can be emptied, then either the cache needs to absorb unlimited modified blocks or eventually the processes modifying the blocks need to be delayed. When a process modifies or adds a block that pushes the total in cache for a particular disk over the limit, the cache manager queues one or more writes for the oldest modified blocks for that disk. If no writes can be queued because the disk queue is full, the process is delayed until some of the blocks have been written. This delay is determined dynamically by taking into account the success rate of previous delays in controlling the increase of modified blocks over the target limit. You can set an upper limit on this delay by using the `max_mod_delay` parameter of the `set_cache_param` request of the `analyze_system` subsystem. The default value is two seconds (`2000 ms`). If you set this parameter to a substantially lower value, the cache manager may not be able to enforce modified block limits for disks that are performing badly. For additional information on this parameter, see ‘‘The `max_mod_delay` Parameter’’ on page 13-23.

A delay of this type is always of limited duration, because, in this situation, the process is typically holding locks. If several processes are running, the maximum number of modified blocks in cache may be somewhat greater than the target value specified by the corresponding tuning parameter.

Per-Volume Attributes

You can specify disk optimization attributes and file allocation attributes on a per-volume basis to match your application’s requirements. You can optimize certain disks for fast response, maximum throughput, or balanced usage (the default). You can specify that, on certain disks, file space be allocated eight blocks at a time or in single-block units. The following sections describe how to configure these per-volume attributes:

- “Per-Volume Optimization Attributes” on page 13-8
- “Per-Volume File Space Allocation” on page 13-9

Per-Volume Optimization Attributes

You configure the per-volume optimization attributes by setting the value of the `opt_for` field in the `disks.tin` file as described in the following sections. See the *OpenVOS System Administration: Configuring a System* (R287) manual for information about configuring the `disks.tin` file.

- “Fast Response” on page 13-9
- “Maximum Throughput” on page 13-9
- “Balanced Usage” on page 13-9

Before per-volume optimization attributes were available, disk optimization involved setting a collection of separate tuning parameters. The tuning parameter method is complex and it only works on a system-wide basis. For example, it does not allow you to optimize some disks for fast response and others for maximum throughput. The disk usage feature is easy to use, it provides much more flexibility, and when used, it typically eliminates the need to make the tuning parameter adjustments described in ‘‘Tuning the Cache’’ on page 13-11.

Fast Response

Some applications, such as those in online transaction processing environments, may require fast access to a certain group of files, because they may set an upper limit on the total elapsed time allowed to process a transaction. If it takes longer, the software may time-out or initiate retries. Fast disk response and minimal latency is required to insure optimal performance for such applications.

To configure disks that contain such files for fast response, set the value of the `opt_for` field to `response` in those disks' entries in the `disks.tin` file. For disks optimized in this way, both the cache manager and disk driver use tuning parameter values and algorithms which minimize latency. By default, file copy operations to or from such disks are paced in order to minimize access time for other files on the disk. By optimizing a disk for fast response, you typically impact its ability to achieve maximum throughput.

Maximum Throughput

You can improve the efficiency of some applications, such as those that write log files (or other files to which fast access is not essential), by optimizing the disks that contain such files for throughput.

To configure disks for maximum throughput, set the value of the `opt_for` field to `thruput` in those disk's entries in the `disks.tin` file. For disks optimized in this way, both the cache manager and disk driver use tuning parameter values and algorithms which produce optimal disk throughput at the expense of fast response time.

Balanced Usage

By default, disks which are not explicitly optimized for response or throughput are optimized for balanced usage. You can also explicitly designate disks to be optimized for balanced usage, by setting the value of the `opt_for` field to `balanced` in those disk's entries in the `disks.tin` file.

Per-Volume File Space Allocation

You can configure how file space is allocated on a per-volume basis to match your application's requirements. You can specify whether to allocate file space on a disk eight blocks at a time or one block at a time.

You use the `alloc_for` field in the `disks.tin` file to configure file-space allocation on a per-volume basis as described in the following sections. See the *OpenVOS System Administration: Configuring a System* (R287) manual for information about configuring the `disks.tin` file.

- ''Allocation for Larger Contiguous Blocks'' on page 13-10
- ''Allocation for Efficient Disk-Space Utilization'' on page 13-10

Additionally you can use the `set_file_allocation` command to specify allocation on a per-file basis. If you use `set_file_allocation` with a value other than the default (`0` or `1`), the value it specifies is always used for that file, regardless of the disk's allocation setting. See the *OpenVOS Commands Reference Manual* (R098) for more information about the `set_file_allocation` command.

NOTES

1. The allocation size that is used, regardless of whether it is set on a per-volume or per-file basis, applies to extent-based files only after all static extents are in use. It does not apply to dynamically allocated extent-based (DAE) files.
2. The allocation size specified by the per-volume file space allocation attribute does not apply to files which have been assigned an explicit non-default allocation size. It also does not affect the allocation of index, directory, or indirect blocks.

Allocation for Larger Contiguous Blocks

Setting the value of the `alloc_for` field to `access` in a disk's entry in the `disks.tin` causes file space to be allocated eight blocks at a time. When more disk space is needed for a file on such a disk, and that file has not been explicitly assigned an allocation size, eight contiguous blocks are allocated.

This provides for faster access time and less performance degradation when disks become fragmented than would occur with one-block allocation. However, if there are many small files less than 8 blocks (32 kilo-bytes) in size, it under-utilizes disk space since all files will occupy an integral number of 8-block units.

Using this option guarantees that all files which do not have an explicit non-default allocation size are allocated in eight block units. If sufficient units are not available, the disk appears to be full and new file allocation cannot occur. Blocks for files with a non-default explicit allocation size are allocated until the disk is completely full, and thus contiguous allocation is not guaranteed.

Allocation for Efficient Disk-Space Utilization

Setting the `alloc_for` field to the value `space` in a disk's entry in the `disks.tin` file causes file space to be allocated one block at a time, for more efficient use of disk space. When more disk space is needed for a non-extent file on such a disk, and that file has not been explicitly assigned an allocation size, disk blocks are allocated one block at a time.

This is also the default behavior when the `alloc_for` field is not specified in a disk's entry in the `disks.tin` file.

Tuning the Cache

If you require disk performance optimization beyond that which is described in ‘‘Per-Volume Attributes’’ on page 13-8, you can tune the cache as described in this and the following section. Contact the Stratus Customer Assistance Center (CAC) or your authorized Stratus service representative before you attempt to tune the cache.

Tuning the cache for maximum performance is an iterative process of monitoring cache performance and then tuning cache parameters based on the results of the monitoring. The recommended process for monitoring and tuning the cache is as follows:

1. Use the default values of the tuning parameters. These default values are usually sufficient for most applications. However, if your application is disk-intensive and you notice performance degradation, you should determine the cause of the degradation using the `analyze_system` subsystem and adjust the cache-tuning parameters accordingly.
2. Monitor the cache performance using the `cache_meters` request of the `analyze_system` subsystem. This request provides a number of useful meters and other relevant information relating to disk cache activity.
3. Tune the cache parameters using the `set_tuning_parameters` command and, if appropriate, the `set_cache_param` request of the `analyze_system` subsystem.

You can modify all cache-tuning parameters using these cache-tuning and monitoring tools. Changes made to the tuning parameters take effect immediately, and you can observe the impact of such changes by using the `dump_cache_info` and `cache_meters` requests of `analyze_system`. You can use these cache-tuning tools to perform the following tasks:

- Control the maximum depth of disk queues.

  Adjusting the maximum depth of disk queues allows for optimization of disk utilization for disks configure for balanced usage. The queue should be sufficiently deep to allow the cache manager to keep it non-empty if writes are pending. It should not be so deep that the disk driver requires excess CPU processing to manage the queues. The maximum queue depth is dynamically adjusted downward if disk response time appears to be increasing. It is adjusted back up to the maximum when disk latency is no longer an issue.

  NOTE

  The maximum queue depth and adjustment policy for disks which are configured for fast response usage or maximum throughput usage is fixed and is not affected by tuning parameters.

- Control the maximum backlog of unwritten disk blocks.

  Setting the maximum backlog of unwritten disk blocks limits the potential wait time when a file is deactivated or the system is shut down. With dynamic modified block limits, this serves only as a starting point.

  NOTE

  The modified block limit policy for disks which are configured for fast response usage or maximum throughput usage is fixed and is not directly affected by tuning parameters.

- Adjust cache behavior based on process priority.

  The cache manager allows low-priority jobs to run normally when cache resources are not being taxed, but does not allow low-priority jobs to interfere with the cache demands of higher-priority processes.

You can also use file attributes and opcodes of the `s$control` subroutine to programatically change the cache behavior for selected files and indexes (see "Open Options and File-Oriented Cache Control" on page 13-30). In addition, you can use the `set_open_options` and `set_ram_file` commands to give files permanent attributes that affect the treatment of their blocks in cache (see *OpenVOS Commands Reference Manual* (R098) for more information on these commands.).

The following sections describe how to tune the cache using the `set_tuning_parameters` command and `analyze_system` requests.

- "Using the `set_tuning_parameters` Command" on page 13-12
- "Using the `set_cache_param` Request" on page 13-18
- "Using the `cache_meters` Request for Interactive Monitoring" on page 13-26

Configuring a disk for fast response or high throughput overrides certain tuning parameter settings discussed in the following sections. As a result, those tuning parameters apply only to balanced usage disks. For example, the modified block limit policy for fast response disks is always static and for high throughput disks is always dynamic. Similarly, the disk write queue is fixed for disks with usage other than balanced and is not affected by the corresponding tuning parameter.

Using the `set_tuning_parameters` Command

You can dynamically control amounts of various resources as well as certain time-related values using the `set_tuning_parameters` command. You can view these values issuing the `display_tuning_parameters` command. This section provides an example of determining what parameter(s) to change, and provides detailed information about some arguments of the `set_tuning_parameters` command. This information supplements the `set_tuning_parameters` command description in Chapter 14.

When you tune the cache, you must balance performance improvement (achieved when you increase the cache size, thereby reducing disk I/O) with the potential degradations that can occur if the system runs out of available memory. You can also tune the cache using the `analyze_system` subsystem and/or the OpenVOS commands `display_system_usage` and `list_users`.

In large cache configurations, the cache manager reaches the limit of virtual memory before it reaches the limit of physical memory. In this case, you should make the limit for physical memory higher than the limit for virtual memory, as the benefit of better performance (resulting from the availability of more buffers, due to the higher limit for physical memory) exceeds the overhead required to manage mapping virtual memory to physical memory.

The following sections provide detailed information about using the `set_tuning_parameters` command:

- "Example of a System with Too Little Cache" on page 13-13
- "Determining Modified Block Limits" on page 13-16
- "Adjusting the Grace Time for Writing Modified Blocks" on page 13-17
- "Setting Eviction Policy Time" on page 13-17
- "Setting Disk Queue Limits" on page 13-17
- "Setting File-Oriented Modified Block Limits" on page 13-18

Example of a System with Too Little Cache

The following example presents data for a system that has too little cache. In this example, a benchmark test of a large database application is executing on the module, and the cache is set to a value that is too small. When a system has too little cache, a large number of requests miss the cache and cause a large number of disk I/O accesses. Thus, examine the number of disk I/O accesses. The output of the `cache_meters` request of the `analyze_system` subsystem displays the number of disk I/O accesses in the `Misses` column, as shown later in this example.

To view the current cache-tuning parameters, issue the `display_tuning_parameters` command, as follows:

```
display_tuning_parameters

The tuning parameters on %system#m9 are:
    max events per process                      8704
    cache_manager, max buffers                  419430
    cache_manager, min buffers                  209715
    cache_manager, max virtual pages            64864
    cache_manager, modified grace time          60 seconds
    cache_manager, transient mod grace time     2 seconds
    cache_manager, unreferenced grace time      300 seconds
    cache_manager, referenced grace time        60 seconds
    cache_manager, free grace time              300 seconds
    cache_manager, cache memory percent         60
    cache_manager, max resident percent         50
    cache_manager, min cache priority           0
    cache_manager, disk mod limit               4096
    cache_manager, disk write limit             1024
    cache_manager, cate write limit             32767
    max processes per module                    4096
    recover_disk priority                       1
    scheduler short wait timeout                0.000 seconds
    unused directory timeout                    120 seconds
    max events per task                         8704
    max events per module                       524287
    max_local_devices_per_module                70000
```

To view the cache activity, issue the `cache_meters` request of the `analyze_system` subsystem.

The following sample output is from the same system:

```
as: cache_meters -reset;sleep -seconds 10;cache_meters

Metering time:   0:00:10
                       Hits          Misses        Total
File      Data         0/  0.00%      0/ 0.00%       0/  0.00%
          Indirect 25176/ 85.07%   4418/ 14.93%  29594/ 47.09%
Index     Data         0/  0.00%      0/ 0.00%       0/  0.00%
          Indirect 27102/ 97.49%    698/ 2.51%   27800/ 44.23%
Directory Data      5420/ 99.98%      1/ 0.02%    5421/  8.63%
          Indirect    34/100.00%      0/ 0.00%      34/  0.05%
Totals             57732/ 91.86%   5117/ 8.14%   62849
```

In this example, the `cache_meters` output indicates (in the `Misses` column of the `Totals` row) that 5117 disk I/O misses occurred in 10 seconds (about 512 per second), which is 8.14% of the total. Reducing the number of misses improves performance (that is, the throughput of the current transactions improves) because a transaction will require less time for physical disk I/O. Memory references for all data in a buffer take approximately 50 microseconds, but the disk I/O required to read (or write) that buffer takes approximately 500 times longer. Since I/O time dominates the entire transaction rate, the biggest benefit derives from avoiding disk I/O.

To display the amount of memory that the system is using, issue an `analyze_system` request similar to the following (do not enter the plus sign (+) because it indicates a line-continuation character).

```
as: wired_memory_meters -reset;sleep -seconds 10;wired_memory_me
+ters
```

The following sample output is from the same system:

```
Metering time:   0:00:10

OWNER                      COUNT     MAX    WIRES     UNWIRES
kernel                      5149    5155        0           0
disk cache                 65537   65537        0           0
wired heap                  9400    9400        0           0
comm heap                      8       8        0           0
link                         288     288        0           0
disk                           0      12      177         177
map                         1276    1291        4           0
I/O heap                     512     512        0           0
streams mblk & d               2      42        0           0
streams dma I/O              230     230        0           0
TOTAL                      82442   181      177

mm.n_wired                  7610
mm.n_temp_wired            74832
mm.TOTAL                   82442
```

The output of `wired_memory_meters` indicates that the system is using 82,442 pages of memory (the disk cache is using 65,537 pages). This particular system has 512 MB of physical memory, which is 128K 4K pages, or in decimal, 131,072 (131,072 = (512 * 1024 * 1024)/(4 * 1024)). Since the system has 131,072 pages from a total of 512 MB, the system should have approximately 48,630 free pages.

When a system needs more cache, and free memory is available, increase the cache size by gradually increasing the value of the `-bm_max_buffers` argument, up to the highest possible value. As you change this value gradually, observe the effects. Leave some free memory (that is, do not reduce the amount of free memory to zero).

In modules with large amounts of physical memory, the physical limit derived from the formula (30% + 60%) may be too conservative. If the cache has reached this limit, and if free memory and high rates of disk I/O still exist, increase the physical limit to 80% or 90% (depending on the stability of the workload), and devote the unused resources to the disk cache. Change this limit by using the `-cm_cache_mem_pct` argument of the `set_tuning_parameters` command.

Determining Modified Block Limits

The cache manager places a per-disk limit on modified blocks for all persistent files. When a file is deactivated or its contents are explicitly runout, all blocks must be committed to disk. If millions of blocks were modified at this time, then closing a file or shutting down a module could become a very lengthy operation. In addition, deactivation is a single uninterruptible operation that occurs in kernel mode while holding locks; thus, other operations requiring access to a related directory may also need to be delayed until deactivation is complete. So, a modified block limit is necessary.

With static modified block limitation, limits are enforced based on a value that you can set using the `-cm_disk_mod_limit` argument of the `set_tuning_parameters` command. This value (`4096`, by default) is applied to each logical disk-pair (or LUN). This single value applies to all disks, and by necessity this value is conservative. For disks configured for fast response usage, the static limit is always used to control modified blocks.

With dynamic modified block limitation, the cache manager determines modified block limits dynamically for each disk by observing the disk's recent and long term performance. Therefore, the actual limit applied is constantly changing, with a target value of the maximum number of blocks that can be written in a set amount of time. For disks configured for maximum throughput usage, the dynamic limit is always used to control modified blocks.

Applications do not typically close or runout files to disk on code paths where performance is critical (for example, where processing delays can lead to timeouts). The static default number of 4096 blocks can often be written to disk within a few seconds, but depending on the layout of the file on disk and other ongoing activity, it could take as much as 10 to 20 seconds to write all modified blocks to a disk. This amount of time applies to the blocks for all files on that disk, so it is hard to predict the maximum amount of time that may be needed to deactivate any particular file.

The default target time for dynamic limits is 30 seconds, which can be achieved within a tolerance of about 20%. Although dynamic limits should serve to improve overall throughput, it can result in the time needed to deactivate a file being noticeably longer. You can use the `set_cache_param` request of the `analyze_system` subsystem to adjust the target time or to choose which limitation policy to use for disks configured for balanced usage. By default, such disks use the `normal` mod limit policy which is the same as static. A hybrid *advisory* policy is also available in which the dynamic limit is used when newly modified blocks are introduced into cache (that is, at points where a process may be stalled) and the static limit is used as a target goal by the `cmt` when determining how many blocks to queue for writing. Typical worst-case deactivation delays of 60 or even 90 seconds are tolerable for most situations, and setting the target to such a higher value will often lead to significantly better performance.

Adjusting the Grace Time for Writing Modified Blocks

You can use the `-bm_modified_grace_time` argument to pace the writing of modified blocks on a time-referenced basis. To minimize the number of write operations to the disk, a block should not be written immediately after being modified if there is a good chance it will be modified again in a short time. This argument, set by default to 60 seconds, defines the maximum amount of time that the cache manager waits for additional modifications before it writes the block to disk. In other words, the cache manager writes a block to disk if it has not been modified in the amount of time specified by the `-bm_modified_grace_time` argument. If the block is being continually modified, the cache manager writes the block to disk in four times the amount of time specified by this argument.

Setting Eviction Policy Time

The cache manager evicts a block from the cache based strictly on the last time it was referenced in combination with other demands on the cache. A modified block cannot be evicted until it is written; however, no relationship exists between how often a block is written to disk and how long it stays in cache. The `cmt` runs asynchronously and evicts blocks that have been unreferenced for a specified period of time (set with the `-bm_unreferenced_grace_time` argument) and writes modified blocks that have not been committed to disk when they exceed the time limit specified by the `-bm_modified_grace_time` argument.

Setting Disk Queue Limits

When the cache manager decides to write a block to disk, it adds a request to the associated disk queue. Disk queues are not processed in a First-In-First-Out (FIFO) manner, but are internally ordered for optimal access based on block address. In order to prevent overloading the disk driver, the cache manager limits the number of writes it has outstanding for each disk. For disks configured for balanced usage, the disk-write limit is controlled by the `-cm_disk_write_limit` argument (of the `set_tuning_parameters` command), which is set to 1024 by default. A non-settable fixed limit is applied to disks which are configured for fast response usage or maximum throughput usage.

If the `cmt` reaches this limit while queueing writes and a disk queue falls to a settable percentage (the `replenish_pct` parameter) of this value, the `cmt` process is immediately rescheduled, in order to continue adding more requests. For optimal disk utilization, the queue should be deep enough to allow for replenishing before it becomes empty. For information on how to set this percentage, see "Using the `set_cache_param` Request" on page 13-18 (particularly the subsection "The `replenish_pct` Parameter" on page 13-25).

Setting File-Oriented Modified Block Limits

The `-cm_cate_write_limit` argument provides a method of limiting modified block usage on a file basis. It is essentially obsolete, and its default value of 32,767 (which disables its effects) should not be changed.

Using the `set_cache_param` Request

The `set_cache_param` request of `analyze_system` enables you to modify certain parameters that control cache characteristics. The `set_tuning_parameters` command enables you to set common tuning parameters that allow straightforward adjustment of cache behavior, as described previously in this chapter. You can use the `set_tuning_parameters` command to change cache-manager behavior (and other aspects of the system) on remote modules as well as on the current module. In some cases it takes special action to assure that the modified tuning values become effective immediately.

In contrast, the `set_cache_param` request of the `analyze_system` subsystem enables you to patch (that is, change the value of) some of the more esoteric parameters that affect cache behavior on only the current module. Values that you modify using the `set_cache_param` request take effect the next time the `cmt` executes. In some circumstances, patching parameters may be effective in improving performance.

NOTE

You should use the `set_cache_param` request only with assistance from the CAC.

The `list_cache_params` request of `analyze_system` lists the values and the current settings of the parameters that the `set_cache_param` request controls.

The following sections provide information about using the `set_cache_param` request.

- "Example of the `list_cache_params` Request" on page 13-19
- "Controlling the `cmt` Run Interval" on page 13-19
- "Limiting Expirations Per `cmt` Run" on page 13-20
- "Setting Modified Block Limit Policy" on page 13-22
- "Resetting Maximum Value Meters" on page 13-24
- "Changing Other Settings" on page 13-24

Example of the `list_cache_params` Request

The following example shows the settable cache parameters and their typical values.

```
as:  list_cache_params

Cache Parameters:

min cm timer interval                (min_cmt_sec)       1 sec
max cm timer interval                (max_cmt_sec)       5 sec
mod limit policy                     (mod_limits)        normal
max_mod limit_delay [1-4000]         (max_mod_delay)     2000 ms
max phys to free per interval        (n_free_expire)     512
max ref expires per interval         (n_ref_expire)      16
max unref expires per interval       (n_unref_expire)    20480
max inmem expires per interval       (n_inmem_expire)    51200
max mod expires per interval         (n_mod_expire)      unlimited
reset max meters [yes]               (reset_maxes)
free phys reserve pct [0-50]         (free_reserve_pct)   2 %
block mode preread amount            (preread_amount)     128
max dae blocks to preread            (preread_ext_max)    128
max dae data blks to preread         (preread_data_max)   64
max dae index blks to preread        (preread_index_max) 16
disk replenish percent [0-99]        (replenish_pct)     50 %
expedited io [y/n]                   (expedited_io)      yes
max pct of cache modified [1-100]    (cache_mod_pct)     85 %
max pct of cache for ram [0-100]     (cache_ram_pct)     80 %
diagnose high latency [y/n]          (diag_latency)      no
max flush time allowed [10-600]      (max_flush_time)    30 sec
index retention time [0-600]         (retain_index_time)  15 sec
```

Controlling the `cmt` Run Interval

The `cmt` is the cache-manager timer process, which runs at a set interval under normal circumstances. (For additional information on the `cmt`, see ‘‘Cache-Manager Timer’’ on page 13-6.) The `cmt` runs at a different, smaller interval the next time when it cannot complete all expirations that it needs to complete on its current execution.

You can change these intervals (defined in terms of seconds) using the `set_cache_param` request to specify values for the `max_cmt_sec` and `min_cmt_sec` parameters. The `cmt` schedules itself to run at the minimum interval when expirations based on grace time are pending, due to hitting one of the defined limits for expiration processing. You can set these limits, too (see ‘‘Limiting Expirations Per `cmt` Run’’ on page 13-20). If modified block expirations are pending due to hitting disk queue limits, the cache manager’s post routine schedules the `cmt` to execute when the queue has been sufficiently emptied. In order to prevent the `cmt` from thrashing, the `min_cmt_sec` parameter specifies the minimum amount of time between iterations of the `cmt` process.

For the `min_cmt_sec` parameter, the range of values is `0` to `60` (seconds); its default value is `1` sec.

For the `max_cmt_sec` parameter, the range of values is `1` to `60` (seconds); its default value is `5` sec.

Limiting Expirations Per `cmt` Run

The `cmt` traverses various lists each time it runs, looking for phys to expire. Expiration is usually associated with grace times, but may also occur due to resource conditions, such as memory emergencies, cache size modification, and detection of excess modified blocks in cache. Expiration of a free phy occurs when the physical memory page reserved for it is returned for general system use; expiration of an unreferenced phy occurs when the phy becomes eligible for freeing; expiration of a modified phy occurs when the associated block is written to disk. Changing cache parameters, such as the percent of memory allocated for memory-resident blocks or the transient modified grace time, causes the `cmt` to traverse lists, making appropriate adjustments.

To prevent excess CPU utilization, the `cmt` limits itself when expiring phys, and you can adjust these limits, as the following sections describe.

- "Free Expiration" on page 13-20
- "Referenced Expiration" on page 13-21
- "Unreferenced Expiration" on page 13-21
- "Memory-Resident Expiration" on page 13-21
- "Modified Expiration" on page 13-22

Free Expiration

The `n_free_expire` parameter sets the number of free phys that are to be returned for general system use when the number of phys exceeds the minimum allowed for the cache and the phy has not been used for 5 minutes, by default. You can set this time value by using the `-bm_free_grace_time` argument of the `set_tuning_parameters` command. The actual maximum and minimum number of phys allowed is based on a percentage of physical memory available, and the `set_tuning_parameters` command enables you to set the maximum and minimum values only within that range (although that percentage can be changed as well).

The default of 512 is intentionally small, to ensure that the amount of memory that is returned to the system is small enough so as not to disrupt cache performance. In cases of memory emergencies, the `cmt` runs more often, at intervals set by the `min_cmt_sec` parameter. If you modify this or similar limits using the `set_cache_param` request of the `analyze_system` subsystem, the new values take effect the next time the `cmt` executes.

Referenced Expiration

Expiring referenced phys involves attempting to notify the process holding the reference to release it thereby changing the phy to the unreferenced state. This occurs if a phy has been last referenced more than one minute ago. You can set this time value by using the `-bm_referenced_grace_time` option of the `set_tuning_parameters` command. Referenced expiration can also occur prior to the grace time criteria. For example, when the cache is very small and a large number of processes are executing, most of the phys in the cache can be currently referenced. In this very unusual circumstance, referenced phys are expired. This is so the resulting unreferenced phy can be evicted from cache, allowing it to be reused for a new request. The `n_ref_expire` parameter sets the maximum number of currently referenced phys that may be expired; its default value of 16 is small.

Unreferenced Expiration

Unreferenced phys may be expired for various reasons. Grace time expiration occurs when a phy has been unreferenced for more than 5 minutes. You can set this time value by using the `-bm_unreferenced_grace_time` argument of the `set_tuning_parameters` command. In addition, when the last opener of a file closes it, all unreferenced phys associated with this phy are expired. Also, certain access patterns (for example, when a single opener of a sequentially accessed file adds new blocks to it) cause blocks to be immediately expired when no longer referenced. In addition, unreferenced phys may be expired prior to the grace time criteria if all cache is in use or the cache reserve is depleted. (See ''The `free_reserve_pct` Parameter'' on page 13-24.)

Expired unreferenced phys either become *free*, if the corresponding file is closed or if expiration is due to the cache reserve, or *outbound*, if it is open. phys on the outbound list are expired to the free list when cache usage is above the specified minimum size, making their associated physical memory page available to return to the system. Both free and outbound lists are treated nearly identically (for purposes of phy and vir), except that outbound phys maintain the identity (and contents) of the disk block they once tracked, whereas free phys do not.

The `n_unref_expire` parameter sets the maximum number of unreferenced phys that may be expired in one `cmt` interval. The default maximum number of unreferenced phys which will be expired in one `cmt` interval is 20,480.

Memory-Resident Expiration

A limited percentage of the cache is allowed for memory-resident files. This percentage of cache space is not reserved for memory-resident files; instead, memory-resident files can occupy a number of phys that is limited to this percentage of cache space. All phys for such files remain permanently in the cache, at least until space is needed for more recent phys from such files. In this case, when space is needed, all phys are treated according to the normal expiration rules. However, the amount of cache for memory-resident files can change. A control operation can immediately change the file attribute from memory-resident to transient. Also, the `-cm_max_resident_pct` argument of the `set_tuning_parameters` command can dynamically decrease the percentage of cache allowed for memory-resident files. When the amount of cache for memory-resident files changes, memory-resident expiration is required because the special treatment of the phys (or a sufficient percentage of phys) for memory-resident files must satisfy the new criteria. The `n_inmem_expire` parameter sets the maximum number of phys allowed for memory-resident files that may be expired.

Modified Expiration

Expiration of a modified phy involves writing the corresponding block to disk. Modified phys are sometimes expired immediately after becoming unreferenced (for example, when one writer is appending blocks to a sequentially accessed file, or when a file is given the transient attribute). But, typically, modified phys are expired based on a grace time that is set using the `-bm_modified_grace_time` argument (default of 60 seconds) of the `set_tuning_parameters` command. The rule is that if a block is not modified for a period of time that is longer than this grace time, the block is expired (that is, the cache manager forces the block to disk) when the `cmt` runs; regardless of how often the block is modified, it will be forced to disk when a period of time that is three times the modified grace time elapses (except for blocks contained in RAM files).

Modified expiration can also occur prior to grace time criteria. When a file with modified blocks is deactivated (closed by the last remaining opener), all such blocks are written immediately. When a disk has more modified blocks than allowed by its limit or if the cache contains more total modified blocks than allowed by the `cache_mod_pct` parameter (see "The `cache_mod_pct` Parameter" on page 13-26), the least recently modified blocks are expired. The `cmt` handles this type of expiration, and no limit exists to the number of phys that can be expired. No limit is necessary because, as the cache manager queues writes of modified blocks, the limit on the disk queue typically causes the `cmt` to cease further mod expiration. If the `cmt` cannot expire several eligible modified phys, the `cmt` schedules itself to run again at the interval set by the `cmt_min_secs` parameter. However, the `cmt` is also scheduled to run when the number of outstanding writes queued to any disk that has reached its mod expiration limit decreases to half the maximum number of disk writes allowed or to any non-default percentage specified by the `replenish_pct` parameter of the `set_cache_param` request of the `analyze_system` subsystem. In this situation, the cache manager's post routine schedules the `cmt`.

The `n_mod_expire` parameter sets the maximum number of phys that may be expired, based on the grace time.

Setting Modified Block Limit Policy

The `cmt` attempts to enforce the modified block limit, but the `cmt` cannot prevent the cache from filling up with modified blocks, since processes adding such blocks can execute faster than the `cmt` can write the blocks to disk. Therefore, the `cmt` works with other components of the cache manager to control unlimited growth. When a process attempts to add a modified block that causes the modified block limit for a particular disk to be exceeded, the cache manager queues disk writes in order to avoid violating the limit. However, because the disk queue also has a limit, queuing disk writes may cause a delay on the process attempting to modify the block. This delay must also be limited, to prevent undue latency for any one process. Ideally, all processes adding new modified blocks should be delayed slightly, to balance their needs with cache resources.

To adjust the algorithm that limits modified blocks, use the parameters of the `set_cache_param` request that the following sections describe:

- "The `mod_limits` Parameter" on page 13-23
- "The `max_mod_delay` Parameter" on page 13-23
- "The `max_flush_time` Parameter" on page 13-24

The `mod_limits` Parameter

Limit modified blocks for disks configured for balanced usage by setting the `mod_limits` parameter to one of the following values:

- The value `static` limits modified blocks according to the value set by the `-cm_disk_mod_limit` parameter of the `set_tuning_parameters` command.
- The value `none` does not limit modified blocks and imposes no delay on the process modifying the block.
- The value `dynamic` enforces the modified block limit dynamically. The target limit is established individually for each disk, using feedback on short-term and long-term performance.
- The value `advisory` uses both static and dynamic limits. The dynamic limit is used when modified blocks are added to cache and the static limit is used by the `cmt` when queueing writes.
- The value `normal` (the default) is the same as the value `static`.

Disks configured for fast response usage always use the static limit, while disks configured for maximum throughput usage always use the dynamic limit, regardless of how this parameter is set. The value `strict`, allowed with earlier OpenVOS and VOS releases, maps to the current value `static`.

The `max_mod_delay` Parameter

The `max_mod_delay` parameter sets the maximum amount of time any process is delayed because the process added an excess number of modified blocks into the cache. Values are `1` to `4000` milliseconds; the default is `2000`.

Setting the `max_mod_delay` parameter to a value that is too low can cause excess modified blocks in cache, resulting in long delays when files are deactivated or when the module is shutdown.

The `max_flush_time` Parameter

The `max_flush_time` parameter sets the target maximum time allowed to write all modified blocks. This value is used to compute per-disk dynamic modified block limits, and it affects the time required to shutdown a module, to remove a disk, and to deactivate a file. Values are `10` to `600` seconds; the default is `30`.

Resetting Maximum Value Meters

The cache manager keeps meters that track maximum values. These meters are unlike normal meters that are controlled by the `-reset` argument of various requests that provide metering information (for example, the `cache_meters` and `disk_lock_meters` requests). The `-reset` argument saves the current meter values and then resets the values of the meters to zero; the saved values are used later to calculate changes in values. In contrast, the `reset_maxes` parameter of the `set_cache_param` request permanently resets the values of the cache-manager meters to zero. The parameter does **not** save values, and the reset values cannot be recovered. However, resetting meter values without saving previous values can be useful when changing cache-manager parameters and observing the results, when, for example, you are attempting to optimally tune the cache for specific situations.

The default value of the `reset_maxes` parameter is `yes` (that is, specifying this parameter resets the values).

Changing Other Settings

The following sections describe additional parameters that you can set using the `set_cache_param` request of the `analyze_system` subsystem.

- "The `free_reserve_pct` Parameter" on page 13-24
- "The `preread_amount` Parameter" on page 13-25
- "The `replenish_pct` Parameter" on page 13-25
- "The `expedited_io` Parameter" on page 13-26
- "The `cache_mod_pct` Parameter" on page 13-26
- "The `cache_ram_pct` Parameter" on page 13-26

The `free_reserve_pct` Parameter

The `free_reserve_pct` parameter specifies the percentage of cache that the cache manager attempts to keep free if all cache is in use. If the cache is full, any new cache usage requires use of a free phy. If all phys are in use, the cache manager forces a phy to be freed by searching lists and, if necessary, resuming the `cmt` to write modified blocks. Attempting to keep a free reserve results in better system performance when the cache is full. Earlier releases of OpenVOS provide the `free_reserve` parameter, which specifies a number of blocks (the default value is 1024), whereas the `free_reserve_pct` parameter specifies a percentage of current cache size. The range of values for the `free_reserve_pct` parameter is `0` to `50`; its default value is `2` for 2%.

The `preread_amount` Parameter

The `preread_amount` parameter sets the number of blocks that are preread during operations of the `copy_file` command. Its values are 0 through 1024; 128 is the default. By setting this parameter to its higher values (up to about 512), you can significantly decrease the amount of time required to copy a file whose blocks are allocated mostly sequentially. This parameter applies only when a block-mode copy is possible, as when, for example, the value of the `-keep_extents` argument (of the `copy_file` command) is `yes` for extent files, or when options that require record-by-record copying are not set (for example, when the value of the `-truncate` argument is `no`). If the `read_ahead` open option is set for the source file of the copy operation, the number of blocks preread during copy operations in block mode is the maximum of the value set for the `read_ahead` open option and for the value of the `preread_amount` parameter.

You can set the `read_ahead` open option using the `-read_ahead` argument of the `set_open_options` command. For information, see "Open Options and File-Oriented Cache Control" on page 13-30 as well as the description of the `set_open_options` command in the *OpenVOS Commands Reference Manual* (R098).

The `replenish_pct` Parameter

The cache manager's goal is to maintain a constant stream of disk-write requests, as long as blocks in the disk cache need to be written, as determined by grace time and the other disk-tuning factors discussed in this chapter. The cache manager checks write completions for each logical unit number (LUN) in an ftScalable Storage system.

The `replenish_pct` parameter sets a percentage of the maximum depth allowed for the queue of outstanding write requests. When the depth of the queue is reduced to this percentage, the cache manager's timer routine is activated, which exposes additional write requests to the disk driver. The default value is `50` for 50%. With a normal queue depth of 1,000 to 2,000 disk-write requests, this value typically prevents the queue from becoming empty. Thus, the disk driver always has a selection of write requests, which it attempts to execute in an optimal fashion.

You may want to change the default value for different reasons. If you specify the `-status` and `-show latency` arguments of the `cache_meters` request of the `analyze_system` subsystem, you can view queue activity and check, at various time intervals, how often the disk queue becomes empty. When the queue becomes empty often, but disk writes are needed, the disk is not being used efficiently. In this case, you may want to increase the `replenish_pct` value so that the disk receives new disk-write requests before the queue is reduced to half empty.

Conversely, it is possible that the efficient queuing of write requests can lead to latency in servicing reads and writes to blocks contained on other areas of the disk. This can be especially true when flushing large amounts of blocks sequentially which are assigned to contiguously ascending disk addresses, as might happen with `copy` or `move_file` operations. Some users may want to use the disk less efficiently in order to avoid the longer latencies. Using `set_tuning_parameters` to reduce the disk queue to a value which is so low that it is impossible to keep the queue from becoming empty is one method to deal with this, but this hurts overall disk performance even in situations where there is no potential for latency. A better method is to keep a reasonable queue depth, but set the `replenish_pct` low enough to allow the queue to often become empty, thereby reducing the chances of high latency.

The `expedited_io` Parameter

The `expedited_io` parameter causes the disk driver to expedite and promptly complete certain types of reads and writes. Writes to indirect and directory blocks often need to be coordinated, which requires waiting for their completion. Such I/O is expedited by bypassing the disk driver's normal algorithm that orders I/Os by disk block addresses. This results in faster response for critical reads and writes. Values are `yes` (the default) and `no`. Setting this value to `no` might cause a significant performance degradation and should be done only under the direction of the CAC.

The `cache_mod_pct` Parameter

The `cache_mod_pct` parameter allows you to change the percentage of cache that can be modified. This parameter provides an upper limit to the number of modified blocks and it takes precedence over other cache parameters. Values are `1` through `100`. The default value is `85%`.

The `cache_ram_pct` Parameter

The `cache_ram_pct` parameter allows you to change the percentage of cache that can be used for modified blocks in RAM files. Blocks which are modified when this limit is reached are treated normally in regards to eviction policy. Such a limit prevents the cache from being exclusively used for RAM files. Setting the limit to `0` essentially eliminates any performance benefit related to RAM files.

Using the `cache_meters` Request for Interactive Monitoring

When modifying cache related parameters, you should observe how the changes affect cache performance. To do so, set up execution patterns that will be sustained over several minutes while you make the changes, and use the `-status` argument of the `cache_meters` request of `analyze_system`. This request provides a convenient summary of many aspects of cache behavior that you can interactively monitor as system load patterns change and as you change tuning or cache parameters. If you specify the `-interval` argument without the `-status` argument, the `cache_meters` request displays a running total of cache-meter values at various intervals, so you can observe values during the last 1, 10, 30, 60 (and so on) seconds.

When you issue the `cache_meters` request with the `-interval` and the `-status` argument (or only the `-interval` argument), the request, by default, displays the data continuously, redisplaying it the number of seconds specified by the `-interval` argument. If the data is logged to a file, specify the `-scroll` argument.

For an example of output of `cache_meters -status`, see *OpenVOS System Analysis Manual* (R073).

The following example shows typical data for the `cache_meters -interval` request, after, in this case, more than an hour has elapsed. Note that with the value `10` for the `-interval` argument, the `cache_meters` display refreshes itself every 10 seconds; what follows is a single such display:

```
as: cache_meters -interval 10
    Cache Meters        metering time: 3:49:57

                10sec   30sec   1min    5min    10min   1hr     Total

Read count      233     782     863     2401    3042    542k    4566k
  ms/read       41.47   39.96   62.28   122.4   211.1   4.501   2.731
  ms/read(l)    64.31   132.0   187.7   279.3   414.3   44.75   27.51
  pct deferred  48.3%   52.6%   57.7%   36.6%   50.8%   44.6%   49.7%
  pre-reads     273     274     280     1383    1383    301k    2294k
  unrefed       1       1       837     837     837     324k    2280k
Write count     2907    8989    18k     94k     877k    1407k   5386k
  ms/write      3.555   3.387   3.402   3.189   0.671   1.647   1.556
  ms/write(l)   3,652   12s     12s     13s     2,733   5,089   2,915
  ref expires   55      55      78      91      258     1511    5132
  unref expires 2700    2700    4587    6143    6437    64k     2287k
  forced        0       0       1912    3472    3926    4526    5218
  delayed       1821    1821    1888    1946    2299    4267    562k
  ms/wdelay     3,772   3,772   4,780   5,808   7,427   7,774   44s
  rel phy       2       2       5       16      59      1698    9195
  mod limit     48      2707    9638    84k     867k    1338k   2900k
Get phys        2534    3157    20k     912k    5249k   8666k   25982k
  ms/get        20.70   48.30   14.58   0.869   0.305   1.205   1.663
  pct hit/miss  79.9%   82.1%   96.7%   99.8%   99.9%   96.9%   90.3%
  hits          2025    2594    19k     910k    5097k   7815k   21334k
  misses        509     563     649     1113    1756    246k    2282k
  pct missed    20.0%   17.8%   3.2%    0.1%    0%      2.8%    8.7%
  soiled        66      86      1230    68k     1101k   2201k   7372k
  pct soiled    2.6%    2.7%    6.1%    7.4%    20.9%   25.3%   28.3%
Rel phys        4846    12k     686k    1575k   4987k   7769k   25041k
  ms/rel        0.549   0.320   0.015   0.289   0.167   0.473   0.274
  modified      3823    10k     677k    1555k   4944k   6706k   15863k
  unmodified    592     653     1009    10k     12k     576k    4811k
```

(Continued on next page)

```
  mod checks     266     356     2756    130k    760k    1416k   4045k
  mod delays     266     356     2752    129k    421k    943k    1552k
  ms/mdelay      10.04   10.04   4.898   3.463   2.014   4.425   5.187
Get free phys    527     578     710     6370    999k    2562k   11318k
  reused         527     578     660     660     660     85k     1655k
  failed         97      184     318     318     318     318     25k
  suspends       0       0       0       0       0       0       0
CMT count        72      126     186     232     290     1559    31k
  ms/exec        55.89   188.9   268.3   255.0   240.7   59.40   7.888
```

The output displays a subset of the values displayed with the `cache_meters -all` request. Additional data is displayed with the `-interval` request if `-all` is specified; non-zero data is displayed if `-long` is specified. The `Total` column is always displayed and represents the total values over the metering time, in this case approximately 3 hours and 50 minutes; this column will be affected by the `-reset` option, but all others are based on the time since the `cache_meters` request was issued, or more accurately, on the last of the various time intervals from the time of the display.

Additional Tuning Issues

The following sections discuss additional issues regarding the tuning of the disk cache. They include discussions on the following topics:

- "Controlling Disk Latency Using the `set_disk_param` Command" on page 13-28
- "Open Options and File-Oriented Cache Control" on page 13-30
- "Process Priority and the Cache" on page 13-35

Controlling Disk Latency Using the `set_disk_param` Command

The disk subsystem, by default, processes disk I/O requests in the order dictated by the number of the required block (called the *cscan ordering optimization*) and not in FIFO order. Sometimes the cscan ordering optimization leads to delays of earlier requests for blocks (that is, requests that are continually being preempted while the cscan ordering optimization satisfies requests for more convenient blocks).

Although the cache manager limits the number of requests queued at one time, its primary goal is to never let a disk queue become empty as long as unqueued write requests are pending. Achieving this goal can lead to a situation in which requests for consecutive ascending disk block addresses can be added to the queue faster than they can be satisfied, and this situation can continue for a relatively long period of time. During this time, a request to read a block from a different area of the disk can stall, and a process waiting for such a read may be significantly delayed.

You can avoid problems of this sort by turning off the cscan ordering optimization and forcing the disk queues to be processed in FIFO order. To do this, use the following `analyze_system` request:

```
set_disk_param cscan no
```

Unfortunately, this change is not recommended since it can result in significant degradation of disk performance, particularly in that it disables the ability to consolidate multiple requests for contiguous blocks. A better way to deal with latency issues is to use the following requests:

```
set_disk_param max_read_latency msec
set_disk_param max_write_latency msec
```

These requests specify the maximum number of milliseconds that a read or write request will be allowed to exist on the disk queue before the ordering optimization is sacrificed and the oldest pending request is honored. This limit is not precise, but normally the time taken to deliver a read or write back to a waiting process will be within one second or so of the specified limit. Typically, processes do not wait for writes to complete; however, you should note that if duplexed writes are being performed serially, the time to complete a logical write requires access to the two partnered disks, and thus write time, as seen by the cache manager and reported with the `cache_meters -status` command, can be up to twice the time specified for the `max_write_latency` parameter.

The default values are set to `500` and `4000` msec, respectively. If read latency is not an issue in your application, increasing this limit may lead to better disk performance. If latency can be sacrificed, then overall access efficiency improves. The default values, however, produce very good results in most typical situations. Setting these to very low values has the effect of defeating the disk cscan optimization algorithm, essentially forcing requests to be processed in FIFO order, but with additional processing overhead.

NOTE

The latency parameters do not affect disks that are configured for fast response usage or maximum throughput usage, since latency control for such disks is always appropriate to their usage.

Open Options and File-Oriented Cache Control

Open options are characteristics of cache behavior that you can associate with a file or index as permanent attributes. OpenVOS automatically applies these attributes when a file is first activated, and the attributes remain in effect until the file is deactivated (that is, when the last accessing process closes the file). Whenever a file is opened, the cache manager treats the file blocks as specified.

The cache-behavior characteristics involve cache mode, read-ahead behavior, and preread extent policy. *Cache mode* refers to how blocks from a file or index are treated in regards to the cache manager's eviction policy. Designating a file as memory resident or transient allows you to override the default eviction policy for blocks of that particular file. Other open options allow you to similarly affect read-ahead behavior and treatment of dynamically allocated extent (DAE) files.

You can use various methods to associate such open options with files and indexes. Certain commands set open options as permanent attributes of a file or its indexes. In addition, you can set directories with default open options, so that files contained in such directories inherit the open options. The commands that set open options are `set_open_options`, `set_default_open_options`, `display_open_options`, and `display_default_open_options`. For information on these commands, see the *OpenVOS Commands Reference Manual* (R098).

Open options apply only during the time a file is activated. A file is activated when it is first opened, and it is said to be active during the time that at least one process has it open. When the last process that has a file opened closes it, it is said to be deactivated, and all its blocks are flushed to disk and removed from the cache. OpenVOS provides a mechanism using the `s$control` subroutine that enables you to programmatically modify certain open options using a port number. In addition, other system subroutines enable you to modify either the permanent open options attributes of any file or the open options that apply to the current activation only. For information on the `s$control` subroutine, see the OpenVOS Subroutines manuals.

The following sections provide additional information about open options:

- "Cache Modes" on page 13-30
- "Read-Ahead Values" on page 13-33
- "PreRead Extent Policy" on page 13-34

Cache Modes

When blocks are read from or written to disk, the cache manager uses values of various tuning parameters to determine which blocks are to be reused for other disk blocks. These values can be overridden by using explicit open options. The blocks of files designated with the transient attribute are evicted quickly from cache and are the most likely to be reused. The blocks of files with the memory-resident attribute are the least likely to be reused. Memory-resident blocks stay in cache until the portion of the cache reserved for memory-resident files is filled. After that, as each new memory-resident block is added, one block that is already memory resident will have its cache mode reduced to normal.

The following sections discuss ways to control the cache by designating memory-resident portions of cache and by setting cache-related file attributes.

- “Memory-Resident Cache” on page 13-31
- “Memory-Resident Files” on page 13-32
- “Transient Files” on page 13-33

Memory-Resident Cache

You can use of the `-cm_max_resident_pct` argument of the `set_tuning_parameters` command to designate a portion of the cache to be used exclusively to hold blocks from the data and/or index portion of one or more files. If the cache is large enough to hold all blocks in a file, the entire file can be accessed directly from cache and is always available to applications, without incurring subsequent disk reads. If there are more file blocks than can fit in this portion of the cache, then the most recently accessed blocks are kept in memory, using the same algorithms that govern other cache activity.

The portion of the cache you reserve for memory-resident files is also used for other purposes, as long as it is not completely filled with blocks from memory-resident files. However, once blocks from memory-resident files have completely filled it, the blocks are never evicted (except to make room for more recently accessed blocks from other memory-resident files).

As discussed previously, processes modifying many blocks in cache are throttled if the number of blocks exceeds a limit. Such processes accessing memory resident files are given favorable treatment over normal files. The dynamic limit based on disk performance is usually substantially greater than the default static limit of 2048. Modified blocks in the memory resident portion of cache are first to take advantage of this overage. Thus, it is possible to acquire the benefits of a RAM file to a limited extent for a normal persistent file by giving it the memory resident attribute and by specifying the value `advisory` or `dynamic` for the modified block limitation policy.

For example, if you have a 1 GB file (244,000 blocks) and you want immediate access to all blocks in that file, you could define a maximum cache size of 1.25 GB (305,175 blocks), and use the `set_tuning_parameters` command to change the `-cm_max_resident_pct` argument from its default of 50% (152,587 blocks) to 80% (244,140 blocks), thus allowing the entire file to be resident. If the file is 2 GB (488, 281 blocks), then the most recently referenced 50% of its blocks remain in cache. See the description of the `set_tuning_parameters` command for more information on this tuning parameter.

Memory-Resident Files

You can designate a file as *memory resident* using the `set_file_cache_modes` file I/O operation of the `s$control` subroutine to set a temporary file attribute. This temporary attribute must be established after the file is open, and it persists until the last process accessing the file closes it. Establishing a file as memory resident does not cause its blocks to be read into memory. Blocks that are never referenced in such a file are never brought into memory.

In addition, you can designate a file as having the memory-resident attribute; such files are always treated as memory resident when they are activated. You can specify a file as memory resident using either a command or subroutine interface.

Control of memory residency is flexible and completely dynamic. You can change the characteristics of a file any time after it is open, and the change in how the file's blocks are treated in the cache is immediately effective. In addition, you can change the percentage of the cache reserved for memory-resident files at any time, and that change is effective immediately.

A typical use of memory-resident files is when you know the contents of a particular file are often referenced and you want those blocks to always be immediately available. If these blocks are constantly referenced, the natural behavior of the cache would be to keep them in memory in preference over other less active blocks. However, consider a situation where at a given point in the day, activity suddenly begins that requires access to a great many blocks that were previously dormant. An example might be the opening of the stock market where the file in question contains stock information. Normally, blocks from this file need to be read into memory, one by one, and delays can occur. Moreover, with much other disk activity going on, even very active blocks may be evicted and need to be reread from disk because by default, every file has equal access to all blocks in the cache.

If you designate such a file and its indexes (that is, blocks that store information about where data blocks are located) as being memory resident and read its blocks into memory before the time of heavy activity, you can specialize the cache so that it is suited for your own particular application. Note, however, that the cache manager evicts all of a file's blocks from the cache and commits them to disk when no process has the file opened.

Cache is never wasted since files not designated as memory-resident files also use memory-resident cache if memory-resident files are not using it. If there are more blocks than can fit in the memory-resident portion of the cache, normal cache competition for blocks occurs.

Using this memory-resident technique can significantly improve throughput, since you can precisely tailor which blocks receive preferential treatment.

Transient Files

The cache manager employs a number of heuristics related to file type and access mode to determine how to treat blocks from various files. For example, a sequential file opened for output has its blocks specially treated to remain in cache a very short time (since the likelihood of them being referenced again is small). The heuristics are quite complex and also involve the number of openers of the file.

Typically, an indexed file opened for random access has no such special treatment for quick eviction for obvious reasons. However, it is not uncommon that such files are used to log daily activity. In this case, records are written sequentially every time a transaction occurs. Often such files roll over into new files when they reach their maximum size. The data in such records may typically not be accessed again for a long time, until it is time to review or audit the transactions of the day. This type of activity has the effect of sweeping the cache. That is, the cache manager thinks these blocks coming into the cache are just as important as the ones already there, and since they were accessed more recently, it evicts existing blocks, which may be the ones containing other information required for every transaction.

You can use the `set_file_cache_modes` file I/O operation of the `s$control` subroutine to designate such files as *transient*. The cache manager evicts a transient file's blocks very quickly. A block from a transient file remains in the cache just long enough for the block to be committed to disk. This minimizes eviction of other blocks already in the cache. (For information on the `s$control` subroutine, see the OpenVOS Subroutines manuals.)

You can specify a file as transient using either a command or subroutine interface. Such files are always treated as transient when they are activated.

When a process writes blocks sequentially, a trailing process often reads them and performs additional processing to them. If the trailing process is too late for a transient file, every block that was just in the cache has to be reread from disk. The `-cm_transient_mod_grace_time` argument of the `set_tuning_parameters` command allows you to avoid this problem by specifying a minimum time for which blocks from a transient file remain in cache after being modified. See the `set_tuning_parameters` command description in Chapter 14 for information on using this tuning parameter.

Read-Ahead Values

Most applications need to read in some percentage of the blocks they use. The performance of these applications improves significantly if the cache manager reads in the blocks before the application uses them. You can use the `read_ahead` open option to override the default behavior. Normally, the file system reads ahead one block, or one extent in the case of a DAE file, when the file is accessed sequentially. The `read_ahead` open option allows you to change that behavior and to either avoid prereads or increase the number of blocks that the file system reads ahead. You can set the read_ahead open option using the -read_ahead argument of the set_open_options command.

When a file is known to be allocated in a more-or-less sequential block order on disk, increasing the value of the read_ahead open option can significantly improve sequential read performance. Setting this value to zero has the same effect as using the s$control PREREAD_MODE operation with flags set to never preread.

The read_ahead open option normally applies to blocks, and except for DAE files, it applies to extents. You can change this behavior using the -preread_extents argument of the set_open_options command (see "PreRead Extent Policy" on page 13-34).

For related information, see "The preread_amount Parameter" on page 13-25. For complete information on the set_open_options command, see the *OpenVOS Commands Reference Manual* (R098), which also describes the related commands set_default_open_options, display_open_options, and display_default_open_options. For information on the s$control subroutine, see the OpenVOS Subroutines manuals.

PreRead Extent Policy

One of the significant factors in the time it takes to read data from a disk drive is head movement. For files that are not extent files, logically adjacent blocks for the file may be assigned physical disk addresses that are far apart. In contrast, with extent files, groups of logically adjacent blocks (up to the size of the extent) are allocated with physically contiguous disk-block addresses. Since reading contiguous blocks requires no head movement or only minimal head movement, it is more beneficial to read the other blocks in an extent at the same time as the first block requested. Due both to disk consolidation (where requests to read or write multiple contiguous blocks can be packaged as a single request) and the on-disk read cache, it is usually very efficient to read multiple, contiguously allocated blocks, as this can often be done without significantly more cost than reading just one block.

However, a tradeoff exists. The additional blocks may increase cache pressure. Therefore, if the blocks are typically never accessed, avoiding prereads, especially if the extent size is large, may improve cache utilization. Note however, a block will not be preread if it requires evicting another cache block, and unreferenced preread blocks will tend to be the first to be evicted, if the need arises. Setting the value of the -preread_extents argument of the set_open_options command to never suppresses any additional blocks from being read from an extent. This has the same effect as using the s$control PREREAD_MODE operation with flags set to avoid extended prereads.

The default behavior (the value 1 for the -read_ahead argument of the set_open_options command and the value normal for the -preread_extents argument) when reading a block in an extent file in random mode is to read that block and the remaining blocks in the extent; when reading in sequential mode, it is to read those blocks plus all blocks in the next extent.

You may wish to control this behavior on a per-file (or per-index) basis. You can use the `OPEN_OPT_PREXT_SEQ_ONLY` open option of the `s$set_open_options` subroutine to change this behavior, so that prereading blocks in an extent occurs only when a file is being accessed sequentially. Note that since index blocks are not read in sequential access mode, this value is not meaningful for them and is considered the same as `OPEN_OPT_PREXT_NORMAL`. You can set this value to never preread (`OPEN_OPT_PREXT_NEVER`) or to `OPEN_OPT_PREXT_FULL`, which causes all blocks in an extent to be preread rather than just those ahead, and which may be useful in optimizing access to index blocks.

For a DAE file, setting the value of the `-read_ahead` argument of the `set_open_options` command to `0` and leaving the value of the `-preread_extents` argument at its default (`normal`) avoids prereading the next extent (which may not be contiguous on disk), but still allows prereading the remaining blocks in the extent, which are guaranteed to be contiguous. Conversely, setting the value of the `-preread_extents` argument to `never` causes the value of the `-read_ahead` argument to be treated in terms of blocks and not extents; thus, essentially treating an extent file as though it were a non-extent file. In order to avoid all prereading in an extent file, you need to set both the value of the `-preread_extents` argument to `never` and the value of the `-read_ahead` argument to `0`.

Currently, the preread policy applies only to DAE files and indexes (that is, it does not apply to statically allocated extent (SAE) files and indexes).

For complete information on the `set_open_options` command, see the *OpenVOS Commands Reference Manual* (R098), which also describes the related commands `set_default_open_options`, `display_open_options`, and `display_default_open_options`. For information on the `s$set_open_options` subroutine, see the OpenVOS Subroutines manuals.

Process Priority and the Cache

Processes can be assigned a priority. Normally, priority is associated with CPU-time allocation. The algorithm for determining scheduling characteristics, and thus the amount of CPU time a process can consume relative to other processes, is determined by the priority assigned to that process. OpenVOS is very flexible in allowing the specification of a number of different scheduling aspects to be associated with a priority number. In general, lower priorities are associated with lower priority numbers.

In some circumstances, it may be useful to control cache access based on process priority. For example, suppose you want to run a background job that uses CPU time when it is available, but defers to other processes when they need the CPU. You can use the `set_priority` command to set a lower priority than normal for the process. OpenVOS allows you to control access to the cache in a similar way, by using the `-cm_min_cache_priority` argument of the `set_tuning_parameters` command. This parameter specifies the lowest priority that gets normal access to the cache. The default setting is 0, which means that no special action is taken for any priority. See the description of the `set_tuning_parameters` command in Chapter 14 for more information on this tuning parameter. See *OpenVOS Commands Reference Manual* (R098) for a description of the `set_priority` command.

If you change the priority to 2, processes with priority 1 or 0 are limited in obtaining access to the cache. Blocks from such processes would never remain in the cache if other blocks are required to be in the cache. This may result in such processes running slower due to the need to do more disk I/O as well as lack of CPU time. When no other demand is placed on the cache or on the CPU, then such a process runs normally with no adverse effects. Blocks referenced by multiple processes are handled according to the priority of the process having higher priority.

RAM Files

A file with RAM usage (known as a *RAM file*) is a file whose storage area is provided mostly by memory. RAM usage allows a file to be stored in memory without any permanent disk storage. You can then use the OpenVOS file-system interfaces to efficiently access this storage area.

RAM files are non-persistent; that is, their contents are valid only while the file is activated. However, such files can take full advantage of available physical memory. Such files can serve as a temporary work area (created by `s$get_temp_file`, for example), providing the file is kept activated (that is, has at least one opener) for the duration of its use.

RAM files have no limit on size, except the limit imposed by the OpenVOS file system. The memory used to access RAM files is taken from the disk cache, and thus is not limited by virtual address space. RAM files require backing store on disk, and if they grow too large to fit in cache, or if there are conflicting demands on cache, blocks are saved to disk. However, unlike with other files, there is no limit to the amount of RAM file data that can exist in cache in a modified state. Blocks of RAM files are written as a background activity and normally writes occur with low frequency and only when the disk is not busy with other activity.

The cache manager places a per-disk limit on modified blocks for all persistent files. Since RAM files are truncated when deactivated, modified block limits are not applicable to them. With regard to modified blocks, files for which the `s$delete_file_on_close` subroutine is called receive the same treatment that RAM files receive, once that call is made.

The following sections provide additional information on RAM files:

- "Using Physical Memory and Virtual Memory" on page 13-37
- "Other Uses of RAM files" on page 13-37
- "Memory Residency and RAM Files" on page 13-38

Using Physical Memory and Virtual Memory

The cache manager can make use of physical memory, sharing a smaller set of virtual memory addresses to access it. As discussed previously, you can set both the size of physical cache and the number of virtual addresses to use with the `set_tuning_parameters` command. If sufficient physical memory is available, the cache manager can easily manage RAM files of 8-10 GB without ever incurring disk reads or throttling program execution due to disk writes, thereby providing a significant performance advantage for temporary work files.

Other Uses of RAM files

Applications using work files that are deleted after use can achieve a performance benefit by calling the `s$delete_file_on_close` subroutine. This subroutine causes the file to be treated as a RAM file except that, on deactivation, the file is deleted rather than truncated. Once this subroutine is called, blocks for the specified file are unlimited in cache and do not count against the maximum allowed for other files on that disk, thereby improving general performance. Such work files typically exist in memory without incurring reads or writes. If your application already calls the `s$delete_file_on_close` subroutine, you get this benefit automatically from the point when the call is made. If your application opens and closes a work file, thereby deactivating it, and then calls `s$delete_file_on_close` only on the last open, you should modify the application so that an additional opener makes the call initially and holds the file open (and, thus, activated) throughout, in order to gain this benefit.

Even when a permanent file is required, an application can achieve significant gains by building the file in memory as a RAM file and then running it out to a permanent file using the `s$copy_file` subroutine. This is because the disk I/O that would be throttling the building of the file could be random, while the disk I/O involved in a copy operation is typically a write to contiguous disk blocks with ascending addresses. Such writes can be performed faster than random writes by easily a factor of 10.

The current limitations of virtual addresses on OpenVOS may affect certain applications that may need very large in-memory data structures. An application can use a fixed RAM file to emulate an in-memory array by calling the `s$rel_write` and `s$rel_read` subroutines to access elements. Although this is obviously significantly more expensive than direct memory access, CPU saturation is not the gating factor for typical applications and this additional overhead is often not prohibitive, providing actual disk I/O never occurs. A relative file can be used if array elements may be null or empty or may contain variable length data. For example, it is possible to implement an in-memory linked list requiring more than 2 GB as a relative file using record numbers in place of pointers. Additionally, one or more indexes can be added to the file to access any array element (i.e., record) directly using a value, possibly self-contained, rather than the element number.

RAM files typically would often be defined with dynamically allocated extents (DAE) since this allows any OpenVOS file types, except stream files, to exceed 2 GB in size. RAM files may or may not be pre-allocated, assuring sufficient disk space for planned usage. If a RAM file has pre-allocated disk blocks when activated, it retains the number of disk blocks that are in use at the time of deactivation, assuring sufficient space for a similar usage; otherwise, its disk blocks are released and are reusable after deactivation.

Memory Residency and RAM Files

It may be useful to use RAM files even when their total size exceeds cache size because you will avoid throttling as the file is being built, with disk writes performed in the background, as needed. In this case, it can be useful to selectively make a specific RAM file memory resident, guaranteeing to the extent possible that its blocks will not be evicted and therefore never incur reads.

Chapter 14 Commands

This chapter documents administrative commands and system process commands.

Administrative commands are specified directly by system administrators. Most commands are of this type. This chapter documents the following administrative commands.

`accounting_admin`	`maint_request`
`add_default_library_path`	`make_message_file`
`analyze_system`	`notify_hardware_error`
`batch_admin`	`process_broadcast_queue`
`broadcast`	`rsn_mail`
`check_jiffy_times`	`rsn_mail_janitor`
`config_console_server`	`set_date_time`
`create_batch_queue`	`set_default_library_paths`
`create_os_symtab`	`set_default_time_zone`
`create_pm_var_file`	`set_jiffy_times`
`delete_default_library_path`	`set_lock_wait_time`
`display_lock_wait_time`	`set_scheduler_info`
`display_scheduler_info`	`set_system_log_mode`
`display_software_changes`	`set_tuning_parameters`
`display_software_purchased`	`site_call_system`
`display_tuning_parameters`	`start_site_call_xfer`
`dump_accounting_file`	`unload_kernel_program`
`list_default_cmd_limits`	`update_codes`
`list_default_library_paths`	`update_default_cmd_limits`
`list_kernel_programs`	`update_program_module`
`load_control_admin`	`update_rsnip_site`
`load_control_histogram`	`update_software_purchased`
`load_kernel_program`	`validate_hub`
`log_syserr_message`	`vtm_config`

System process commands are rarely or never used outside of `module_start_up.cm` or other `.cm` files. These commands usually start service processes for the module.

`batch_overseer`	`start_rsn`
`monitor_rsn_servers`	`tp_overseer`
`network_watchdog`	`wait_for_overseer`
`overseer`	`wait_for_tp_overseer`

The operator key sequences in this chapter are for the V105 terminal with the PC/+ 106 keyboard and the V105/V109 with the EPC keyboard. The OpenVOS Software Release Bulletin provides the corresponding key sequences for the V105 terminal with the ANSI keyboard.

`accounting_admin` ***Privileged***

Purpose

The `accounting_admin` command enables or disables the logging of statistics for a module and specifies the type of statistics to be logged.

Display Form

```
------------------------------ accounting_admin ----------------------------
module:                    █
-disable_accounting:       no
-port_accounting:          no
-log_commands:             no
-log_files:                no
-log_proc_stats_records:   yes
-log_proc_user_records:    yes
-log_transactions:         no
```

Command-Line Form

```
accounting_admin [module]
            [-disable_accounting]
            [-port_accounting]
            [-log_commands]
            [-log_files]
            [-no_log_proc_stats_records]
            [-no_log_proc_user_records]
            [-log_transactions]
```

Arguments

▶ *`module`*

The module for which the accounting facility is being enabled or disabled. The default is the current module.

▶ `-disable_accounting` CYCLE

Disables accounting on the specified module. When disabling accounting, do not use this argument with any other argument, as new settings are ignored. When enabling accounting, set this argument to `no` to retain the other values. The default value is `no`.

▶ `-port_accounting` CYCLE

Records statistics about the I/O traffic on each port on the designated module. The default value is `no`. This argument is automatically set to `yes` when the `-log_files` argument is set to `yes`. Port accounting degrades system performance slightly.

▶ `-log_commands` CYCLE

Logs command-start and command-termination records for each command executed. The default value is `no`. Command logging degrades system performance noticeably.

▶ `-log_files` CYCLE

Logs a file-close record whenever a file is closed. When this argument is on, the `-port_accounting` argument is set to `yes`. The default value is `no`. File logging degrades system performance noticeably.

▶ `-no_log_proc_stats_records` CYCLE

Does not log a process statistics record whenever a process calls the subroutine `s$log_resource_usage`. The default value is `yes`.

▶ `-no_log_proc_user_records` CYCLE

Does not log a process-defined record whenever a process calls the subroutine `s$log_process_record`. This allows you to create user-defined log records. The default value is `yes`.

▶ `-log_transactions` CYCLE

Logs a transaction record whenever a transaction starts, commits, or aborts. The default value is `no`. Transaction logging can degrade system performance noticeably, depending on the number of transactions.

Explanation

The `accounting_admin` command enables or disables statistic-logging for a module. When accounting is enabled, process statistics and user-defined log records are logged by default. Commands, port-use records, file-use records, and transactions may also be logged.

Examples

The following command enables port accounting and command logging on module `m5`, but does not keep a process statistics record.

```
accounting_admin m5 -port_accounting -log_commands
                -no_log_proc_stats_records
```

The following command logs the closing of any file on the current module and the starting, committing, or canceling of any transactions on the current module. This command disables the writing of process-defined records by OpenVOS.

```
accounting_admin -log_files -no_log_proc_user_record
                -log_transaction
```

The following command disables accounting on module `m1`.

```
accounting_admin m1 -disable_accounting
```

Related Information

See Chapter 8, "The Accounting Facility," for definitions of the types of accounting records.

add_default_library_path *Privileged*

Purpose

The `add_default_library_path` command adds a path name to the list of directories that define the specified library. The new library paths are in effect for all new processes on a module.

This command is used mainly in the `module_start_up.cm` file.

Display Form

```
--------------------------- add_default_library_path -------------------------
library_name:          include
library_path_names:
-before:
-after:
-first:                no
-check:                yes
-module:               current_module
```

Command-Line Form

```
add_default_library_path [library_name]
            library_path_names
            [-before existing_library_path_name]
            [-after existing_library_path_name]
            [-first]
            [-no_check]
            [-module module_name]
```

Arguments

▶ *`library_name`* CYCLE

The name of a library to which the path name of a directory is to be added. Possible values of *`library_name`* are `include`, `object`, `command`, and `message`. The default value is `include`.

▶ *`library_path_names`* **Required**

The path name of one or more directories. You can specify the path name of any of these libraries by using one of the command functions `'(current_dir)'` or `'(home_dir)'`; if *`library_name`* is `message`, the path name can include the command function `'(referencing_dir)'` or `'(language_name)'`. The command function must be enclosed in apostrophes.

NOTE

You can specify only one of the following three arguments (`-before`, `-after`, or `-first`). If you omit all three of these arguments, the `add_default_library_path` command adds the directory to the end of the list.

▶ `-before` *`existing_library_path_name`*

Specifies that *`library_path_names`* is to be inserted before *`existing_library_path_name`* in the library list.

▶ `-after` *`existing_library_path_name`*

Specifies that *`library_path_names`* is to be inserted after *`existing_library_path_name`* in the library list.

▶ `-first` CYCLE

Places *`library_path_names`* at the beginning of the library list.

▶ `-no_check` CYCLE

Does not check that each of the values specified for the *`library_path_names`* argument is a valid object name. The default value is `yes`.

When you include this command in the `module_start_up.cm` file, specify the `-no_check` argument. Otherwise, if an object named as a library path name is on another system, the startup fails because the object is not known to the module.

▶ `-module` *`module_name`*

Specifies the module whose list of library paths is to be changed. If you omit this argument, OpenVOS assumes the current module.

Explanation

The `add_default_library_path` command allows you to add one or more path names anywhere in the list of directories that define a specified library for a module.

A *library* is a set of directories that OpenVOS searches for objects of a particular type. Each library is defined by an ordered sequence of path names. There are four libraries.

- the `include` library, in which the compilers search for include files
- the `object` library, in which the binder searches for object modules
- the `command` library, in which the command processor searches for commands
- the `message` library, in which the command processor searches for per-command message files and help files

The module's default list of directories for each library serves as the guide for finding objects. The search for an object begins in the first directory on the library list. If the object is not in that directory, the search proceeds to the second directory on the list, and so on.

The `add_default_library_path` command gives you control over which directories OpenVOS searches for an object. It enables you to insert a path name anywhere in the list of directories for a given library. Use the `-before` argument to determine where in the list to place the additional path name.

The path name of the directory *`library_path_names`* can include either the command function `(current_dir)` or `(home_dir)`; if the value of the *`library_name`* argument is `message`, the path name can also include the `(referencing_dir)` or `(language_name)` command function. (Enclose a command function in apostrophes in order to prevent its evaluation by the command processor when you issue it with the `add_default_library_path` command.)

The list of libraries modified by the `add_default_library_path` command remains in effect for the module until it is changed by any of the following privileged commands.

```
add_default_library_path
delete_default_library_path
set_default_library_paths
```

Examples

Suppose you issue the command `list_default_library_paths include` and the output shows that the module's `include` library consists of the following directories.

```
include library directories:
     (current dir)
     %s1#d03>Sales>incl
     %s1#d04>system>include_library
```

To add a directory called `tools` to the set of `include` library paths, execute the following command.

```
add_default_library_path include tools -before
>system>include_library
```

The system expands the relative path name. Now, if you issue the command `list_default_library_paths include`, the output is as follows:

```
include library directories:
     (current dir)
     %s1#d03>Sales>incl
     %s1#d03>Sales>tools
     %s1#d04>system>include_library
```

Related Information

See the following commands in this manual: `delete_default_library_path`, `set_default_library_paths`, and `list_default_library_paths`. See descriptions of the `add_library_path`, `delete_library_path`, `list_library_paths`, and `set_library_paths` commands and the `(current_dir)`, `(home_dir)`, `(referencing_dir)`, and `(language_name)` command functions in the *OpenVOS Commands Reference Manual* (R098). See the *OpenVOS Commands User's Guide* (R089) for information about OpenVOS's search rules.

analyze_system *Privileged*

Purpose

The `analyze_system` command controls access to the analyze-system subsystem, which is a diagnostic tool that you can use to display information either about a dump or how OpenVOS is functioning on a module.

Display Form

```
-------------------------------- analyze_system -----------------------------
-request_line:  █
-module:
-quit:          no
-c_expressions: no
```

Command-Line Form

```
analyze_system [-request_line string]
            [-module module_name]
            [-quit]
            [-c_expressions]
```

Arguments

▶ `-request_line` *`string`*
Specifies a subsystem request or two periods followed by an OpenVOS internal command. In most cases, you enter a request. If you specify an OpenVOS internal command, it must be preceded by the two periods (`..`). In the command-line form, if the string contains spaces, apostrophes, or parentheses, it must be enclosed in apostrophes.

If you do not specify a request line as you enter the subsystem, `analyze_system` prompts you with `as:`. A string entered after the `as:` prompt should not be enclosed in apostrophes.

▶ `-module` *`module_name`*
Specifies the module that `analyze_system` is to look at when responding to requests. The default value is the current module.

▶ `-quit` CYCLE

Returns the process to command level. Normally, you would use this argument with a `-request_line` argument so that `analyze_system` will return you to command level immediately after performing the request specified in `-request_line`.

To exit the subsystem at a later time, issue the `quit` request.

▶ `-c_expressions` CYCLE

Specifies that internal expressions be evaluated using a C language line parser instead of the traditional `analyze_system` line parser. The C language line parser implements a subset of the keywords and commands supported by the traditional `analyze_system` line parser. For information about expression evaluation, see the *OpenVOS System Analysis Manual* (R073).

Explanation

The analyze-system subsystem is a command processing loop. Once you enter the loop, `analyze_system` displays the `as:` prompt whenever it expects you to enter a request or command. To exit the loop and return to command level, issue the request `quit`.

The `analyze_system` command uses the OpenVOS symbol table, which is created each time a new version of OpenVOS is booted. If, while executing `analyze_system`, you get an error message related to the symbol table file, create a new symbol table using the `create_os_symtab` command.

Examples

The following command invokes the subsystem request `dump_et` on module `m12`.

```
analyze_system -request_line dump_et -module m12 -quit
```

The following command invokes the OpenVOS internal command `start_logging` and names the logging file created `as_log`.

```
analyze_system -request_line '.. start_logging as_log'
```

Related Information

See the manual *OpenVOS System Administration: Starting Up and Shutting Down a Module or System* (R282) and the `create_os_symtab` command, described in this manual. For more information about the `analyze_system` requests, see the *OpenVOS System Analysis Manual* (R073).

`batch_admin` *Privileged*

Purpose

The `batch_admin` command tells OpenVOS to either stop or start processing the batch requests in a specified batch queue. It can also specify which module is to run the batch processes in the specified queue and can set the maximum number of batch requests that can be processed at one time from the queue.

Display Form

```
---------------------------------- batch_admin -------------------------------
request:             stop
queue_name:
-module:
-overseer_module:
-max_users:          1
```

Command-Line Form

```
batch_admin request
            [queue_name]
            [-module queue_module]
            [-overseer_module overseer_module]
            [-max_users number]
```

Arguments

▶ *request* CYCLE

An action to be taken on the specified batch queue. The allowed actions are `stop`, `start`, and `stop_all`. The `stop_all` action shuts down all of the batch queues that are running on the Overseer module and, for that reason, cannot be specified with the *queue_name* argument.

- *queue_name*

  The name of the queue on which OpenVOS is to start or stop processing batch requests. The queue name must not be an extended name.

  You cannot issue this argument with the `stop_all` value of the *request* argument. If you specify `stop` or `start` as the *request* argument (or omit *request* in the command-line form) you are prompted for a queue name.

- `-module` *queue_module*

  Specifies the name of the module containing the specified queue. If you omit this argument, OpenVOS assumes the queue is on the current module.

- `-overseer_module` *overseer_module*

  Specifies the module name or module star name that is to run the batch processes in the specified queue. If you omit this argument, the current module will run the batch processes in that queue.

- `-max_users` *number*

  Specifies the maximum number of batch requests that the overseer module can process at one time from the specified queue. The default value is `1`; the maximum value is `256`.

Examples

The following command starts processing, on module `m5`, the batch requests in the queue `batch_2` located on the current module.

```
batch_admin start batch_2 -overseer_module m5
```

The following command starts processing on the current module the batch requests in the queue `batch_1`, located on module `m1`. No more than three batch requests will be processed at the same time.

```
batch_admin start batch_1 -module m1 -max_users 3
```

Related Information

See Chapter 4, ‘‘The Batch Processor.”

`batch_overseer` ***Privileged***

Purpose

This is a system process command. It is used **only** by the Overseer process.

The `batch_overseer` command starts the BatchOverseer process that processes requests in batch queues.

Display Form

```
------------------------------- batch_overseer ------------------------------
No arguments required.  Press ENTER to continue. █
```

Command-Line Form

```
batch_overseer
```

Explanation

The `batch_overseer` command is used only by the Overseer process to start the BatchOverseer process for a module. You should never need to execute this command.

Related Information

See Chapter 4, "The Batch Processor," and the `overseer` command description later in this chapter.

broadcast

Purpose

The `broadcast` command sends a message to all login terminals on either one or more modules or throughout a system.

Display Form

```
---------------------------------- broadcast --------------------------------
message:
-module: ''
```

Command-Line Form

```
broadcast message
          [-module modules]
```

Arguments

▶ *`message`* **Required**

The text of the message to be broadcast. The message must be no longer than a single terminal display line. When you specify the message in the command-line form, it must be enclosed in apostrophes if it contains any spaces.

▶ `-module` *`modules`*

Specifies a module name or star name. To broadcast the message throughout all modules in the current system, specify the value `*` for *`modules`*. If you omit this value, OpenVOS broadcasts the message to users on the current module.

Examples

The following command sends throughout the system the message that is enclosed in apostrophes.

```
broadcast 'System shutting down at 7 PM' -module *
```

The following command sends to all login terminals on module `m2` the message that is enclosed in apostrophes.

```
broadcast '%s1#m2 shutting down at 7 pm' -module %s1#m2
```

Related Information

See the description of the `send_message` command in the *OpenVOS Commands Reference Manual* (R098).

check_jiffy_times *Privileged*

Purpose

The `check_jiffy_times` command displays information about the date and time settings on each module. It is used to check whether the settings are within acceptable tolerances among the modules in a system.

Display Form

```
------------------------------ check_jiffy_times ----------------------------
-time_set: no
-relative:
```

Command-Line Form

```
check_jiffy_times [-time_set]
         [-relative module_name]
```

Arguments

▶ `-time_set` CYCLE

Displays the current date and time on each module in the system, relative to the time on the current module. The default value is `no`.

The `-time_set` argument displays the time that the date and time were last synchronized or set on all the modules in a system.

▶ `-relative` *`module_name`*

Displays the current date and time on each module in the system, relative to the given module name. The default value is the current module.

Examples

The following output is produced by invoking `check_jiffy_times`. The `TIME CHECKED` column shows the time on the current module. The `TIME WAS` column shows the current time on every module in a system. The `DELTA` column shows the difference between the time on the current module, and the time on the other modules in a system.

```
    check_jiffy_times

MODULE        TIME CHECKED              TIME WAS         DELTA
m1   97-11-17 08:48:52.7725  97-11-17 08:48:52.7718  -0.0006 sec
m2   97-11-17 08:48:52.7725  97-11-17 08:48:52.7712  -0.0012 sec
m3   97-11-17 08:48:52.7725  97-11-17 08:47:57.2523  -55.5201sec
m4   97-11-17 08:48:52.7725  97-11-17 08:48:52.7742  +0.0017 sec
m5   97-11-17 08:48:52.7725  97-11-17 08:48:52.7734  +0.0009 sec
m6   97-11-17 08:48:52.7725  97-11-17 08:48:52.7730  +0.0005 sec
```

The following sample output displays the time on `m6` that the date and time were last synchronized on the modules in this system. The `TIME SET` column shows the last synchronization time for the system, according to the clock on module `m6`.

```
    check_jiffy_times -time_set -relative m6

    MODULE          TIME SET
    m1          97-11-17 08:48:17.9373
    m2          97-11-17 08:48:17.9373
    m3          97-11-16 08:48:17.9373
    m4          97-11-17 08:48:17.9373
    m5          97-11-17 08:48:17.9373
    m6          97-11-17 08:48:17.9373
```

Related Information

See the commands `set_jiffy_times` and `set_date_time` later in this chapter.

`config_console_server` *Privileged*

Purpose

The `config_console_server` command assigns a site ID and other information to an RSN dialup console server (that is, a console server that connects to a modem) as well as to the RSN bridge module to which the console server is attached. (If your connection is through the Internet, or if you are configuring either an AA-E97900 RSN console server or an AA-E99100 RSN console server, use the `update_rsnip_site` command.)

NOTE

You **must** contact the CAC before using this command. You **must** coordinate with the CAC any changes that you want to make in argument values.

Display Form

```
------------------------------- config_console_server ---------------------------
console_server_IP:     IP_address
RSN_bridge_IP:
site_id:
site_password:
cac_login:
cac_passwd:
-rsn_in_port:    85
```

Command-Line Form

```
config_console_server console_server_IP
            RSN_bridge_IP
            site_id
            site_password
            cac_login
            cac_passwd
            rsn_in_port number
```

Arguments

- ▶ *`console_server_IP`* **Required**

  The IP address of the console server that you are configuring. Specify a 4-byte IP address in standard dot notation.

- ▶ *`RSN_bridge_IP`* **Required**

  The IP address of the RSN bridge module to which the console server is connected. Specify a 4-byte IP address in standard dot notation.

- ▶ *`site_id`* **Required**

  The site ID of the RSN bridge module. This ID is also referred to as the RSN site ID. Specify a value of up to 16 characters. The value cannot include the characters `%`, `<`, `>`, `!`, or `#`. The value that you specify must be identical to the value specified by the *`site_id`* argument of the `start_rsn` command.

- ▶ *`site_password`* **Required**

  The site password of the RSN bridge module. Specify a value of up to eight characters.

  This value is identical to the password that you specify on the `SITE PARAMETERS` screen of the `maint_request` command's `update` request. When `PASSWORD ENABLED` is set to `yes` on the `SITE PARAMETERS` screen, the RSN HUB must supply this password in order to connect to this RSN bridge module. If *`site_password`* is not identical to the password stored in the RSN HUB, then the console server cannot log in to the RSN HUB.

- ▶ *`cac_login`* **Required**

  The user ID that the console server uses as a login. Note that this is not the OpenVOS user ID or the `rsn_admin` user ID, but it is a backup user ID that the CAC can use to access the system, if the `rsn_admin` user ID is not operational, in some emergency situation. Contact the CAC for this user ID.

- ▶ *`cac_passwd`* **Required**

  The password that the console server uses with the *`cac_login`* user ID. For more information, see the description of the *`cac_login`* argument, above. Contact the CAC for this password.

- ▶ `-rsn_in_port` *`number`*

  The port number that the console server uses to log in to the CAC. Contact the CAC for this number.

Explanation

You issue the `config_console_server` command to assign the site ID and password to an RSN dialup console server (that is, a console server that connects to a modem; if your connection is through the Internet, or if you are configuring either an AA-E97900 RSN console server or an AA-E99100 RSN console server, use the `update_rsnip_site` command) in the following situations:

- Before configuring the RSN bridge server for the first time, while installing the module.
- When you have installed a new console server (for example, when replacing a faulty console server).

NOTE

You **must** contact the CAC before using this command. You **must** coordinate with the CAC any changes that you want to make in argument values.

You issue this command after you have connected the console server to the RSN bridge module, but before you connect the console server to the modem.

Related Information

For more information on the RSN, see Chapter 6. See also the following manuals:

- *OpenVOS STREAMS TCP/IP Administrator's Guide* (R419) for information on the `start_stcp.cm` file
- *OpenVOS System Administration: Configuring a System* (R287) for information about configuring OpenVOS for the RSN
- The site planning guide as well as the operation and maintenance guide for your system (as listed in Related Manuals in the Preface) for information about physically connecting the console server to the RSN bridge module

create_batch_queue

Purpose

The `create_batch_queue` command creates a batch queue on the specified module.

Display Form

```
------------------------------ create_batch_queue ----------------------------
queue_name:
-module:
```

Command-Line Form

```
create_batch_queue queue_name
            [-module module_name]
```

Arguments

- `queue_name` **Required**

  The name of the batch queue to be created. Do not specify an extended name. A batch queue is a file in the directory `(master_disk)>system>queues>batch`.

- `-module module_name`

  Specifies the name of the module to contain the specified queue. If you omit this argument, OpenVOS assumes the current module.

Examples

The following command creates the batch queue `batch_4` on module `m7`.

```
create_batch_queue batch_4 -module m7
```

Related Information

See Chapter 4, ‘‘The Batch Processor.”

`create_os_symtab` ***Privileged***

Purpose

The `create_os_symtab` command creates a version of the OpenVOS symbol table file for the current version of OpenVOS.

Display Form

```
------------------------------ create_os_symtab ----------------------------
-force: yes
```

Command-Line Form

```
create_os_symtab [-no_force]
```

Arguments

▶ `-no_force` CYCLE

Does not create the OpenVOS symbol table. The default value is `yes`. If you specify `-no_force`, the command checks to see if the current symbol table (if it exists) is for the current version of OpenVOS and, if so, disregards the command.

Explanation

The OpenVOS symbol table is created each time a new version of OpenVOS is booted. The symbol table file resides in the directory `(master_disk)>system`. The name of the file is `os.symtab`. If the symbol table is unusable, you get an error message when executing programs that reference the symbol table. If this occurs, you can manually boot the system and create the file with the `create_os_symtab` command. Creating the symbol table file takes about two minutes.

Related Information

See the manual *OpenVOS System Administration: Starting Up and Shutting Down a Module or System* (R282).

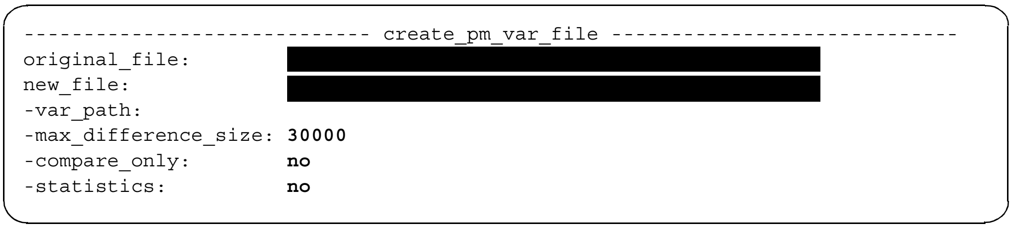
create_pm_var_file

Purpose

The `create_pm_var_file` command compares two OpenVOS program modules and generates a variation file containing the differences. If the number of differences exceeds 30,000 bytes, the system displays the following message.

```
Maximum difference limit of 30000 bytes has been exceeded.
```

Display Form



```
---------------------------- create_pm_var_file ----------------------------
original_file:
new_file:
-var_path:
-max_difference_size: 30000
-compare_only:        no
-statistics:          no
```

Command-Line Form

```
create_pm_var_file original_file new_file
        [-var_path path_name]
        [-max_difference_size size]
        [-compare_only]
        [-statistics]
```

Arguments

▶ *`original_file`* **Required**
The path name of the base program module. Do not specify an extended name.

▶ *`new_file`* **Required**
The path name of the new program module. Do not specify an extended name.

▶ `-var_path` *`path_name`*
Specifies the path name of the variation file. Do not specify an extended name. By default, the command uses the name *`original_file.var`*.

- `-max_difference_size` *`size`*
  Specifies the size (in bytes) of the variation file. By default, the command uses the value 30,000, which is also the file's maximum size.

- `-compare_only` CYCLE
  Compares the program modules but does not make a variation file. Information about the number of blocks in each file and the differences between the files is displayed.

- `-statistics` CYCLE
  Records the amount of clock and CPU time used when running this command and provides information about the distribution of record differences (for example, how many differences fell into a given range).

Explanation

The `create_pm_var_file` command compares two OpenVOS program modules and generates a variation file containing the differences. Depending on the extent to which the program modules differ, the variation file generally will be 4 to 5 times smaller than either *`original_file`* or *`new_file`*. Thus, if the original program module is 100 blocks long, the variation file may be only 25 blocks long. This reduction in size is very useful when distributing software over wide area networks. The variation file is created and saved in the same directory in which `create_pm_var_file` is run. The `create_pm_var_file` command will not process a program module longer than 4,000 blocks.

The size of the variation file is a function of the number and size of differences between the two program modules. Two program modules that vary greatly may produce a variation file very close in size to that of the larger of the original program modules. When the variation file exceeds 85% of the size of the requested program module, program execution stops and the system displays a message to the user.

The length of time it takes the command to run generally depends on the number of differences found between the two program modules. When two files are similar, the files are compared quickly.

The variation file generated is named *`original_file.var`*. For example, if you want to compare `merge_directories1.pm` against `merge_directories2.pm`, the name of the variation file generated is `merge_directories1.var`.

You can use this command in conjunction with the `update_program_module` command to produce revisions of existing software program modules. The `update_program_module` command uses the variation file and the *`original_file`* file to re-create *`new_file`*. You can use the `create_pm_var_file` and `update_program_module` commands with programs generated in VOS Release 7.0 or later.

Examples

The following command compares two versions of the program `dump_database`.

```
create_pm_var_file
%s#m1>installed>commands>pm>dump_database.pm
        %s#m1>development>commands>pm>dump_database.pm
-statistics

Loading program modules...

File A: %s#m1>installed>commands>pm>dump_database.pm
  Size: 8 blocks

File B: %s#m1>development>commands>pm>dump_database.pm
  Size: 9 blocks

Number of diffs: 278    Words of difference: 6333
Number of difference bytes: 14424 (4 blocks)

Percent size reduction: 61

Difference record distribution statistics:

    Size    Adds    Dels    Eqls    Mods
       1       1       3      59       1
       2       4       6      23      11
       5       3       1      42      33
      10       6       2      45      28
      50       0       0       0       5
     100       0       1       0       2
     500       0       0       0       0
    1000       2       0       0       0
    5000       0       0       0       0
    ****       0       0       0       0

              16      13     169      80

Clock Time: 5     CPU Time: 1.560104     Page Faults: 37
```

Related Information

See Chapter 7, ‘‘Updating Program Modules,’’ and the description of the `update_program_module` command later in this chapter.

`delete_default_library_path` ***Privileged***

Purpose

The command `delete_default_library_path` deletes a path name from the list of directories that define the specified library. The changes are effective for all new processes on a module.

This command is used mainly in the `module_start_up.cm` file.

Display Form

```
------------------------ delete_default_library_path ------------------------
library_name:        include
library_path_name:
-module:             current_module
```

Command-Line Form

```
delete_default_library_path [library_name]
            library_path_name
            [-module module_name]
```

Arguments

- ▶ *`library_name`* CYCLE

  The name of a library from which the path name of a directory is to be deleted. Possible values of *`library_name`* are `include`, `object`, `command`, and `message`. The default value is `include`.

- ▶ *`library_path_name`* **Required**

  The path name of a directory. You can specify the path name of any of these libraries by using the command function `'(current_dir)'` or `'(home_dir)'`; if the *`library_name`* value is `message`, you can also include `'(referencing_dir)'` or `'(language_name)'`. The command functions must be enclosed in apostrophes.

► `-module` *`module_name`*
    Specifies the module whose list of library paths is to be changed. If you omit this argument, OpenVOS assumes the current module.

Explanation

The `delete_default_library_path` command allows you to remove a single path name from the list of directories that defines a specified library for a module.

A library is a set of directories that OpenVOS searches for objects of a particular type. Each library is defined by an ordered sequence of path names. There are four libraries.

- the `include` library, in which the compilers search for include files
- the `object` library, in which the binder searches for object modules
- the `command` library, in which the command processor searches for commands
- the `message` library, in which the command processor searches for per-command message files

The module's default list of directories for each library serves as the guide for finding objects. The search for an object begins in the first directory on the library list. If the object is not in that directory, the search proceeds to the second directory on the list, and so on.

The `delete_default_library_path` command allows you to control which directories OpenVOS searches for an object. It enables you to delete any path name in the list for a specified library.

The path name of the directory *`library_path_name`* can include one of the command functions `(current_dir)`, `(home_dir)`, `(referencing_dir)`, or `(language_name)`, enclosed in apostrophes. (Enclose a command function in apostrophes in order to prevent its evaluation by the command processor when you specify the command function with the `delete_default_library_path` command.)

The list of libraries modified by the `delete_default_library_path` command remains in effect for the module until it is changed by any of the following privileged commands.

```
add_default_library_path
delete_default_library_path
set_default_library_paths
```

Examples

Suppose you issue the command `list_default_library_paths include` and the output shows that the module's `include` library consists of the following directories.

```
include library directories:
     (current dir)
     %s1#d03>Sales>incl
     %s1#d03>Sales>Smith>work
     %s1#d04>system>include_library
```

To delete the directory named `work` from the set of `include` library paths, execute the following command.

```
delete_default_library_path include %s1#d03>Sales>Smith>work
```

If you issue the command `list_default_library_paths include`, the output appears as follows:

```
include library directories:
     (current dir)
     %s1#d03>Sales>incl
     %s1#d04>system>include_library
```

Related Information

See the following command descriptions in this chapter: `add_default_library_path`, `set_default_library_paths`, and `list_default_library_paths`. See the *OpenVOS Commands Reference Manual* (R098) for descriptions of the `add_library_path`, `delete_library_path`, `list_library_paths`, and `set_library_paths` commands and for information about the `(current_dir)`, `(home_dir)`, `(referencing_dir)`, and `(language_name)` command functions. See the *OpenVOS Commands User's Guide* (R089) for information about OpenVOS's search rules.

display_lock_wait_time

Purpose

The `display_lock_wait_time` command displays, for each specified module, the default wait time for I/O operations that can return one of the following messages.

```
e$file_in_use          e$no_pipe_readers
e$key_in_use           e$no_pipe_writers
e$record_in_use        e$tp_wait_to_win
e$region_in_use
```

Display Form

```
-------------------------- display_lock_wait_time ------------------------
-module: current_module
```

Command-Line Form

```
display_lock_wait_time [-module module_name]
```

Arguments

- ▶ `-module` *`module_name`*
  Specifies a module name or star name whose wait time is to be displayed. If you omit this argument, OpenVOS defaults to the current module.

Explanation

The `display_lock_wait_time` command displays, for each specified module, the default wait time for certain I/O operations. The *wait time* is the maximum number of seconds that a process or task waits to lock either a file or record during any I/O operation. If no program or process wait time has been set, the default lock wait time is used.

Examples

The following command displays the wait time for implicit locking on all modules of the current system.

```
display_lock_wait_time -module *
```

Related Information

See the command descriptions of `set_lock_wait_time` later in this chapter and of `display_process_lock_wait_time` and `set_process_lock_wait_time` in the TPF manuals. See the *OpenVOS Commands Reference Manual* (R098) for an explanation of locking.

display_scheduler_info

Purpose

The `display_scheduler_info` command displays the current scheduling parameters for interactive and batch processes.

Display Form

```
---------------------------- display_scheduler_info --------------------------
-module: current_module
```

Command-Line Form

display_scheduler_info [-module *module_name*]

Arguments

▶ `-module` *module_name*
Specifies the module for which the current scheduling parameters will be displayed. The default value is the current module.

Examples

The following output of the `display_scheduler_info` command displays the current scheduling parameters for module `%sales#m1`. The column titles in the output are abbreviated. Their full meanings are: `PRI` is the priority level, `SKT` is the socket, `TYPE` is the time-slice type, `CNT` is the count, and `TIME` is the CPU time in seconds.

```
display_scheduler_info -module %sales#m1

--INTERACTIVE--                  -----BATCH-----
PRI     SKT  TYPE   CNT   TIME          SKT  TYPE   CNT   TIME
 0        1     4     3   0.125           1    14     1   0.250
                6     5   0.250
                6     5   0.125
               10     1   0.125
 1        1     4     5   0.250           1    12     1   0.125
                4     5   0.125
                6     5   0.125
               10     1   0.125
 2        1     2     1   0.063           1    12     1   0.250
                6     5   0.250
                6     5   0.125
               10     1   0.125
 3        1     2     1   0.125           1    10     1   0.125
                4     3   0.250
                6     5   0.250
               10     1   0.125
 4        1     2     1   0.125           1    10     1   0.250
                4     3   0.250
                6     8   0.250
               10     1   0.125
 5        1     2     1   0.125           1    10     1   0.375
                4     5   0.250
                6     5   0.250
               10     1   0.125
 6        1     2     1   0.125           1     8     1   0.250
                4     5   0.250
                6     5   0.375
                8     1   0.125
 7        1     2     1   0.125           1     6     1   0.125
                4     5   0.375
                6     5   0.375
                8     1   0.125
 8        1     2     2   0.125           1     4     1   0.250
                4     5   0.375
                6     5   0.375
                8     1   0.125
 9        1     2     3   0.125           1     2     1   0.250
                4     5   0.500
                6     5   0.500
                6     1   0.250
```

Related Information

See Chapter 2, ‘‘Setting and Defining Priority Levels,’’ and the `set_scheduler_info` command description later in this chapter.

display_software_changes

Purpose

The `display_software_changes` command displays the changes that will be made the next time you run the `update_software_purchased` command. The Remote Service Network (RSN) stores information about changes sent from the support center, and these commands enable the module's software-purchased bits information to be updated at the customer's convenience.

Display Form

```
---------------------------- display_software_changes -----------------------
-module: current_module
```

Command-Line Form

```
display_software_changes
            [-module module_name]
```

Arguments

- ▶ `-module` *`module_name`*
  Specifies the name of the module whose list of pending software-purchased bit changes should be displayed. You can also specify a star name. The default value is the name of the current module.

Explanation

The *`display_software_changes`* command displays, for each specified module, the purchased software changes that are pending until you execute the `update_software_purchased` command.

Examples

The following command displays the purchased software changes that are pending on all modules of the current system.

```
display_software_changes -module *
```

Related Information

See the description of the `update_software_purchased` command later in this chapter.

display_software_purchased

Purpose

The command `display_software_purchased` displays the software that you purchased from Stratus and that is available for use on the specified module(s).

Display Form

```
------------------------- display_software_purchased ------------------------
-module:       ''
-table_path:
```

Command-Line Form

```
display_software_purchased [-module module_names ...]
```

Arguments

- `-module` *`module_names`*
  Specifies the module names of the modules about which you want to display the available software. You can specify one module, a list of modules, or a star name as *`module_names`*. A separate list of available software appears for each module you specify. If you omit this argument, OpenVOS displays the software available on the current module.

- `-table_path` *`path_name`*
  Specifies the path name of the `software_purchase_table.table` file. This file, which is located in the `(master_disk)>system` directory, lists the software that you purchased from Stratus and that is available for use on the module.

Explanation

The `display_software_purchased` command displays the Stratus software that is available on the specified modules. It also includes the software product code and the product name.

If you are upgrading to a new release of OpenVOS, the `-table_path` argument enables you to display the status of new software products introduced in the version of

OpenVOS to which you are upgrading. To check on the status of new software products **before** upgrading to a new release of OpenVOS, perform the following steps.

1. Run the verification phase of the OpenVOS installation process.
2. Issue the following command to change the current directory to the new release directory.

   ```
   change_current_dir (master_disk)>system>release_dir
   ```

3. Issue the following command to display the status of new software products introduced in the version of OpenVOS to which you are upgrading.

   ```
   display_software_purchased -table_path
   software_purchase_table.table
   ```

   The system displays the software that you purchased from Stratus and that will be available for use on the module **after** you upgrade to the new release of OpenVOS. If the list of software is incomplete, contact the CAC.

For additional information about the OpenVOS installation process, see the *OpenVOS Installation Guide* (R386).

display_tuning_parameters

Purpose

The `display_tuning_parameters` command displays system tuning parameters for a specified module.

Display Form

```
-------------------------- display_tuning_parameters ------------------------
-module: current_module
```

Command-Line Form

`display_tuning_parameters` [`-module` *`module_name`*]

Arguments

- `-module` *`module_name`*

  Specifies the module whose tuning parameters are to be displayed. If you omit this argument, OpenVOS displays the tuning parameters of the current module.

Explanation

The `display_tuning_parameters` command displays the tuning parameters for the specified module. Note that with the exception of `max processes per module` (see the Examples section), you can modify these parameters with the `set_tuning_parameters` command.

Examples

The following is output from the `display_tuning_parameters` command. The parameters are described following the sample output; the values shown are the defaults for a ftServer module with 4GB of physical memory.

```
The tuning parameters on %s1#m1 are:
    max events per process                   8704
    cache_manager, max buffers               419430
    cache_manager, min buffers               209715
    cache_manager, max virtual pages         64864
    cache_manager, modified grace time       60 seconds
    cache_manager, transient mod grace time 2 seconds
    cache_manager, unreferenced grace time  300 seconds
    cache_manager, referenced grace time     60 seconds
    cache_manager, free grace time           300 seconds
    cache_manager, cache memory percent      60
    cache_manager, max resident percent      50
    cache_manager, min cache priority        0
    cache_manager, disk mod limit            4096
    cache_manager, disk write limit          1024
    cache_manager, cate write limit          32767
    max processes per module                 4096
    recover_disk priority                    1
    scheduler short wait timeout             0.000 seconds
    unused directory timeout                 120 seconds
    max events per task                      8704
    max events per module                    524287
    max_local_devices_per_module             70000
```

Parameter Descriptions

`max events per process`
The maximum number of user events, allowed per process, on the module.

`cache_manager, max buffers`
The maximum number of physical disk cache buffers.

`cache_manager, min buffers`
The minimum number of physical disk cache buffers.

`cache_manager, max virtual pages`
The maximum number of virtual pages, used to map the physical pages, in the cache.

`cache_manager, modified grace time`
The length of time, in seconds, that a modified block is left in memory before it is

written to disk.

`cache_manager, transient mod grace time`
The length of time, in seconds, that the cache manager waits after a block from a transient file is modified before the cache manager writes the block to disk.

`cache_manager, unreferenced grace time`
The length of time, in seconds, before an unreferenced physical page is available for reuse, as a buffer, for a different disk block.

`cache_manager, referenced grace time`
The length of time, in seconds, before a referenced physical page is forcibly unreferenced.

`cache_manager, free grace time`
The length of time, in seconds, before an unreferenced physical page is returned to the pool of physical pages available for virtual memory management.

`cache_manager, cache memory percent`
The percentage of memory over 128 MB to use for the cache.

`cache_manager, max resident percent`
The percentage of the maximum cache size available for memory-resident files.

`cache_manager, min cache priority`
The minimum priority for processes that are entitled to retain data in the cache.

`cache_manager, disk mod limit`
The maximum number of modified blocks per disk that the cache manager is allowed to hold in cache before initiating disk writes.

`cache_manager, disk write limit`
The maximum number of write requests that the cache manager is allowed to queue to each disk.

`cache_manager, cate write limit`
The maximum number of writes that are allowed to be concurrently queued for blocks associated with a cate before delays will occur for processes attempting to subsequently access other blocks in that cate.

> NOTE
>
> This value is no longer used; it appears only for compatibility purposes.

`max processes per module`
The maximum number of processes allowed on the module.

`recover_disk priority`
: The default priority level of any process created to perform a `recover_disk` operation.

`scheduler short wait timeout`
: The number of seconds that OpenVOS keeps a process in memory, pending notification of an event.

`unused directory timeout`
: The number of seconds a directory can be unused before its contents are written to disk.

`max events per task`
: The maximum number of events that a task can wait for.

`max events per module`
: The maximum number of events allowed on the module.

`max_local_devices_per_module`
: The maximum number of devices allowed on the module.

Related Information

You can modify all of the system tuning parameters, except for `max processes per module`, by issuing the `set_tuning_parameters` command, described later in this chapter. For information about optimizing the tuning parameters on your module, see the `(master_disk)>system>doc>cache_tuning.doc` file.

dump_accounting_file

Purpose

The `dump_accounting_file` command displays or writes to a file some or all of the records in an accounting log.

Display Form

```
----------------------------- dump_accounting_file ---------------------------
start_record:        0
stop_record:         32767
log_path:            >system>accounting>raw_acct_log.(date)
-output_path:
-match_process_id:  0
```

Command-Line Form

```
dump_accounting_file [start_record]
            [stop_record]
            [log_path]
            [-output_path path_name]
            [-match_process_id number]
```

Arguments

▶ `start_record`

The first record either displayed or written to a file. The default value is record number `0`.

▶ *`stop_record`*

The last record either displayed or written to a file. The default value is record number `32767`.

▶ *`log_path`*

The path name of the accounting log file to be either displayed or written to a file. Do not specify an extended name. The default path name is `(master_disk)>system>accounting>raw_acct_log.(date)`, where `(date)` is the current date.

- `-output_path` *`path_name`*
  Directs the output to *`path_name`*. Do not specify an extended name. If you omit this argument, OpenVOS displays the records on your terminal.

- `-match_process_id` *`number`*
  Displays only those records with process IDs that match the number specified in this argument.

Explanation

The `dump_accounting_file` command is a useful tool for debugging programs that use accounting software.

Examples

The following command writes to the file `acct_info` the records through record number 500 from the accounting log for the current date on the current module.

```
dump_accounting_file 0 500 -output_path acct_info
```

Related Information

See Chapter 8, ‘‘The Accounting Facility,” and the description of the `accounting_admin` command earlier in this chapter.

list_default_cmd_limits

Purpose

The `list_default_cmd_limits` command lists the initial and maximum default command limits that apply to all processes created on the module.

Display Form

```
---------------------------- list_default_cmd_limits -------------------------
module: current_module
```

Command-Line Form

`list_default_cmd_limits` *`module`*

Arguments

- ▶ *`module`*
  The name of the module for which you want to list the default command limits. By default, the command uses the current module.

Explanation

The `list_default_cmd_limits` command lists the initial and maximum values for heap limit, stack limit, total space limit, CPU time limit, stream file size limit, keep module size limit, and limit on the number of attached ports, which are the module’s default command limits.

Examples

The following example shows sample output from the `list_default_cmd_limits` command.

```
Default command limits for module %sys#m1.
        Initial total limit:  infinity.
        Initial heap limit:   infinity.
        Initial stack limit:  8388608.
        Initial cpu limit:    infinity.
        Initial file limit:   infinity.
        Initial keep limit:   infinity.
        Initial port limit:   4096.
        Maximum total limit:  infinity.
        Maximum heap limit:   infinity.
        Maximum stack limit:  132513792.
        Maximum cpu limit:    infinity.
        Maximum file limit:   infinity.
        Maximum keep limit:   infinity.
        Maximum port limit:   4096.
```

The value `infinity` indicates that the command imposes no limit.

Related Information

For a description of the initial and maximum default command limits, including the units in which they are expressed, see the description of the `update_default_cmd_limits` command.

To specify the initial and maximum limits for any program module executed within an existing process, use the `update_process_cmd_limits` command. In addition, use the `update_process_cmd_limits` command to modify the process command initial and maximum limits. The `update_process_cmd_limits` command is described in the *OpenVOS Commands Reference Manual* (R098).

`list_default_library_paths`

Purpose

The `list_default_library_paths` command displays the list of directories that OpenVOS searches through for the designated library.

Display Form

```
------------------------- list_default_library_paths ------------------------
library_name: all
-module:      current_module
```

Command-Line Form

```
list_default_library_paths [library_name]
         [-module module_name]
```

Arguments

- ▶ `library_name` CYCLE

  The library whose list of default library paths will be displayed. Allowed values are `include`, `object`, `command`, `message`, and `all`. The default value is `all`.

- ▶ `-module` *`module_name`*

  Specifies the module whose library paths will be displayed. If you omit this argument, OpenVOS assumes the current module.

Examples

The following command displays the default path names of all libraries on `m5`.

```
list_default_library_paths -module m5

Module %s1#m5

include library directories:
     (current_dir)
     %s1#d01>system>Oracle_include_library.500
     %s1#d01>system>include_library

object library directories:
     (current_dir)
     %s1#d01>system>object_library

command library directories:
     (current_dir)
     %s1#d01>system>command_library
     %s1#d01>system>tools_library

message library directories:
     (referencing_dir)
     (referencing_dir)>(language_name)
     %s1#d01>system>message_library>(language_name)
```

Related Information

See the following commands in this chapter: `add_default_library_path`, `delete_default_library_path`, and `set_default_library_paths`. See the *OpenVOS Commands Reference Manual* (R098) for descriptions of the `add_library_path`, `delete_library_path`, `list_library_paths`, and `set_library_paths` commands. See also the *OpenVOS Commands User's Guide* (R089) for information about OpenVOS's search rules.

list_kernel_programs *Privileged*

Purpose

The `list_kernel_programs` command lists the path names of the program modules that are currently loaded into the OpenVOS kernel of the current module.

Display Form

```
---------------------------- list_kernel_programs ----------------------------
-all: no
```

Command-Line Form

```
list_kernel_programs
            [-all]
```

Arguments

▶ `-all` CYCLE

The value `yes` specifies that the command also displays the path names of the program modules that are in the process of being unloaded. By default (the value `no`), the command lists only the path names of the program modules that are currently loaded.

Explanation

The `list_kernel_programs` command lists the path names of program modules such as I/O drivers or user-routine libraries that are currently loaded into the OpenVOS kernel.

Kernel-loadable program modules are produced by the `bind` command. The `load_kernel_program` command loads a copy of the program module into the kernel address space.

Examples

The following command lists the path names of the program modules that are currently loaded on the current module, `%eng#m1`.

```
list_kernel_programs -all

Loaded Kernel Programs:

%s1#m1>system>kernel_loadable_library>vterm.pm
%s1#m1>system>kernel_loadable_library>tnmod.cp.pm
%s1#m1>system>kernel_loadable_library>timod.cp.pm
%s1#m1>system>kernel_loadable_library>pt_tioc_mod.cp.pm
%s1#m1>system>kernel_loadable_library>streams_pipe.cp.pm
%s1#m1>system>kernel_loadable_library>udp.cp.pm
%s1#m1>system>kernel_loadable_library>ip.cp.pm
%s1#m1>system>kernel_loadable_library>loop.cp.pm
%s1#m1>system>kernel_loadable_library>stcp.cp.pm
%s1#m1>system>kernel_loadable_library>sdlmux.cp.pm
%s1#m1>system>kernel_loadable_library>genet.cp.pm
%s1#m1>system>kernel_loadable_library>adp.cp.pm
%s1#m1>system>kernel_loadable_library>arp.cp.pm
%s1#m1>system>kernel_loadable_library>tpipe.cp.pm
%s1#m1>system>kernel_loadable_library>sosl_net_driver.cp.pm
```

Related Information

For information about loading and unloading kernel programs, see the descriptions of the `load_kernel_program` and `unload_kernel_program` commands. For information about the `bind` command, see the *OpenVOS Commands Reference Manual* (R098).

load_control_admin **Privileged**

Purpose

The `load_control_admin` command performs the following tasks.

- starts the load control process
- stops the load control process
- causes OpenVOS to recognize an updated load control table during the current bootload

Display Form

```
----------------------------- load_control_admin ---------------------------
request:            login
-login_priority:    0
-overseer_module:
-log:               no
```

Command-Line Form

```
load_control_admin request
            [-login_priority number]
            [-overseer_module overseer_module]
            [-log]
```

Arguments

▶ `*request*` CYCLE **Required**

Three values are allowed for this argument.

- `login` starts the load control process on the specified module.
- `logout` stops the load control process on the specified module.
- `reconfigure` tells OpenVOS on the specified module to read the load control configuration table (`load_configuration.table`), and the load control action table (`load_control.table`). Use this request when you have updated one or both of the tables and you want OpenVOS to recognize the updated table immediately, rather than at the next bootload. See the manual *OpenVOS System Administration: Configuring a System* (R287).

▶ `-login_priority` `*number*`

Specifies the priority of the load control process that will be logged in. This value must be an integer from 0 to 9. It should be the same as the priority of the average module user. The default value for this argument is `0`.

▶ `-overseer_module` `*overseer_module*`

Specifies the module on which to start, stop, or reconfigure the load control process. The default value is the current module.

▶ `-log` CYCLE

Writes load control information to the file `load_control_log.(date)` in the directory `(master_disk)>system`.

Examples

The following command starts the load control process, giving it a priority of `5`, and writes load control information to a log file.

```
load_control_admin login -login_priority 5 -log
```

The following command causes OpenVOS to recognize an updated load control table during the current bootload.

```
load_control_admin reconfigure
```

Related Information

See Chapter 5, ‘‘The Load Control Facility.”

load_control_histogram

Purpose

The `load_control_histogram` command converts the contents of a load control log file into a histogram.

Display Form

```
---------------------------- load_control_histogram --------------------------
source:     >system>load_control_log.(date)
-avg_load: yes
```

Command-Line Form

```
load_control_histogram [source]
            [-no_avg_load]
```

Arguments

- ▶ *`source`*

  The path name of the load control log file. Do not specify an extended name. If you do not specify a file, OpenVOS uses the `load_control_log.(date)` file with the current date. The log files are in the `(master_disk)>system` directory.

- ▶ `-no_avg_load` CYCLE

  Displays the histogram in terms of decisive load factors (averages of the preliminary load factors). The default is `yes`. When you invoke the `-no_avg_load` argument, OpenVOS displays the preliminary load factors. (Chapter 5, “The Load Control Facility,” defines these terms.)

Related Information

See Chapter 5, “The Load Control Facility.” To write a histogram to a file, see the commands `start_logging` and `stop_logging` in the *OpenVOS Commands Reference Manual* (R098).

`load_kernel_program` ***Privileged***

Purpose

The `load_kernel_program` command loads a program module, such as a library of user routines, separately into the OpenVOS kernel. You can issue this command either in the `module_start_up.cm` file, or during the current bootload.

Display Form

```
----------------------------- load_kernel_program -----------------------------
kernel_program:
type:             library
```

Command-Line Form

```
load_kernel_program kernel_program
        [type]
```

Arguments

▶ `kernel_program` **Required**
The path name of the program module to be loaded into the kernel.

▶ *`type`* CYCLE
The type of loadable program module; either `driver` or `library`. The default value is `library`.

N O T E

You typically use the `configure_comm_protocol` command, to load a driver into the system. See the manual *OpenVOS System Administration: Configuring a System* (R287) for information on this command.

Explanation

Using the `load_kernel_program` command to load I/O drivers separately can result in a large savings of wired memory and virtual address space in OpenVOS since you need only load those protocols your system requires. The `load_kernel_program` command allows you to add your own library routines to OpenVOS's kernel, without needing to customize OpenVOS.

Kernel-loadable program modules are produced by the `bind` command. The `load_kernel_program` command loads a copy of the program module into the kernel address space. This command adds designated entry points in the routines in the loadable program module to the list of kernel entry points. User programs can access these entry points in much the same way that programs access traditional kernel routines.

Some kernel-loadable program modules can be unloaded without rebooting the module. For information about unloading kernel-loadable program modules, see the description of the `unload_kernel_program` command later in this chapter.

Kernel-loadable program modules have the following restrictions.

- The code must be re-entrant. For example, for loadable routines written in COBOL, the `mode is recursive` phrase must be used in the `object-computer` statement when the loadable routines are compiled.
- The loadable routines execute in user mode even though they reside in the kernel address space. They cannot modify the kernel and they cannot masquerade as traditional kernel entry points. Thus, the routines offer no advantage or disadvantage from a security or privileged usage aspect. They can call other kernel entry points (privileged or nonprivileged), including ones in the same or another loaded program module. The program modules must be loaded in the correct order if this is done. Privileged entry points must still be called only by a privileged process.
- Variables in the static region of a loadable routine are accessed by all tasks in all processes that use that routine. However, static variables are located in the kernel address space and thus cannot be written (changed) by the code, which runs in user mode.
- You must execute the `load_kernel_program` command to load the loadable routines before you can access them. An exception to this rule is the STREAMS drivers, which are automatically loaded the first time an application opens a device that uses the driver.
- You cannot delete kernel programs or replace the loaded routines by a subsequent `load_kernel_program` command.

- The entries to your routines must conform to the following kernel entry naming conventions.
  - The entry name must start with `s$`, and must not start with `s$$`. To avoid conflicts with present and future kernel entry points, begin the names of your subroutines with `s$u_`, which has been reserved for user loadable routines.
  - You must write the code for these subroutines and follow the instructions for loading programs in the kernel to use these subroutines.
  - The entry name must not already be defined, either in the kernel or in a loadable routine that is already loaded in some other loadable program module.
  - You can only use the characters `a-z`, `0-9`, and _ to name your routines. Do not use uppercase letters.
- The loadable routines must not use language I/O routines because the I/O routines use static variables.
- You cannot use some C-run-time routines, such as `ctime`. The routines are prohibited because they modify a static location and then return a pointer to that location.

In addition to these restrictions, consider the following factors when deciding between kernel-loadable routines and other methods of accessing user-written routines.

- The loadable routines reduce the virtual address space available for the kernel to use. The `analyze_system` request `dump_vm_pool_info` displays information about the use of virtual address space in the kernel.
- You can create versions of the loadable routines that provide the same interface, but internally act differently. For example, suppose you need encryption routines, but your application has to operate on systems both in and out of the United States. By building one version that uses the Data Encryption Standard (DES) method, and another that uses another technique, you can avoid the export licensing requirements associated with DES. The application programs that call the routines do not have to change. See the *OpenVOS PL/I Subroutines Manual* (R005) or the *OpenVOS C Subroutines Manual* (R068) for more information on using kernel-loadable DES subroutines.
- Changes to the loadable routines, even if they need a reboot, can be made to the single (shared) copy.

- The traditional method of binding the routines in each copy of a user program module is as follows:
  - requires duplicated disk space for each copy of each program module
  - reduces the user virtual address space (but it is generally less critical than kernel address space)
  - loads the code only when it is needed (but multiple copies might be loaded if the routine is used in multiple program modules)
  - requires a rebind of every application program module that uses the routines whenever a change needs to be made, but this might be less costly than a reboot
- Any program that calls the loadable routines can be compiled and bound at any time, even before the routines are loaded into the kernel. In this case, the binder issues warnings, but the unresolved references are resolved when the program executes, provided the loadable routines are present at that time.

Examples

Perform the following steps to create a user-loadable routine, load it into the kernel, and call it from another program.

1. Create the loadable routine.

```
/* file_name:  'my_resident_library.pl1' */

dcl s$write        entry (char(*) var);

my_resident_routine_1:
   procedure;

     call s$write ('This is kernel resident routine \#1');
      return;
      end;

my_resident_routine_2:
   procedure(a_message);

dcl a_message       char(*) varying;

      call s$write (a_message || ' <from routine \#2>');
      return;
   end;
```

2. Compile the loadable routines.

```
pl1 my_resident_library.pl1
```

3. Create a binder control file. Name the object modules that contain the routines you need. Do not include searches of object library paths or you might accidentally include library routines that are not needed or wanted. Use the `no_library` argument to suppress the library search. Always specify the `paged` section, and remember that the `load_in_kernel` and `relocatable` arguments are required. Specify the `retain` directive to retain only the necessary entry points. The binder control file appears as follows:

   ```
   /* file_name 'my_resident_library.bind' */

   options: kernel, load_in_kernel, relocatable, no_library;
   name:    my_resident_library;
   section: paged;
   modules: '>my_object_dir>my_resident_library';
   retain:  my_resident_routine_1 as s$u_resident_routine_1,
            my_resident_routine_2 as s$u_resident_routine_2;
   end;
   ```

   NOTE

   The binder control-file options may change depending on which language the program module is written in. The `subroutines_are_functions` option is used to bind C programs.

4. Create the loadable program module.

   ```
   bind -control my_resident_library.bind -map -table
   ```

   Inspect the `my_resident_library.map` file, looking for unintentional inclusion of library routines, retention of unwanted entry points, and use of internal and/or external static variables.

5. Move the loadable program module to a directory that will be accessible during module startup. Do **not** use the `(master_disk)>system>kernel_loadable_library` directory, since it is a controlled system directory; your routines would be removed during the next system installation.

   ```
   move_file my_resident_library.pm>system>user_library
   ```

6. Activate the loadable routines by loading the program module into the kernel. To load `my_resident_library` into the kernel, add the following line to the `module_start_up.cm` file and reboot, or issue it interactively.

   ```
   load_kernel_program >user_library>my_resident_library
   library
   ```

NOTE

When you issue the command interactively, the user-loaded routines are in force only until the next bootload.

When a kernel program is loaded, `load_kernel_program` logs a message to the `syserr_log.(date)` file. Loading `my_resident_library.pm` sends the following message to the `syserr_log.(date)` file.

```
09:43:00 >user_library>my_resident_library.pm loaded by
Joan_Sullivan.Marketing
```

7. To call the now-resident `s$u_` routines, reference them as you would any kernel routine.

```
test_resident_library:
     procedure;

dcl s$u_resident_routine_1 entry;
dcl s$u_resident_routine_2 entry (char(*) var);

  call s$u_resident_routine_1;
  call s$u_resident_routine_2('Test message');

end;
```

8. Compile, bind, and execute the test program.

```
pl1 test_resident_library.pl1
bind test_resident_library.obj
test_resident_library
```

The following messages appear on your terminal.

```
This is kernel resident routine #1
Test message <from routine #2>
```

If you try to execute a program that calls user-loaded routines that you have not loaded, you receive the following error messages.

```
load_program: Warning--the following links cannot be
resolved:
s$u_resident_routine_1
s$u_resident_routine_2
An attempt was made to call a missing module.
```

(Continued on next page)

```
s$u_resident_routine_1
Referencing address 00000001x.
Error occurred in procedure test_resident_library, line 7.
Command was test_resident_library.pm.
Request?  (stop, continue, debug, keep, login, re-enter)
```

Related Information

For information about listing and unloading kernel programs, see the descriptions of the `list_kernel_programs` and `unload_kernel_program` commands in this chapter. For information on the `bind` command, see the *OpenVOS Commands Reference Manual* (R098).

log_syserr_message

Purpose

The `log_syserr_message` command enters a message into the file `syserr_log.(date)` for the current date in the `(master_disk)>system` directory of the specified module. Use this command in the `module_start_up.cm` command file to log any messages generated during the startup.

Display Form

```
------------------------------ log_syserr_message ----------------------------
message:
-module:
-direct:  no
-console: yes
```

Command-Line Form

```
log_syserr_message message
           ⎡-module module_name  ⎤
           ⎣      -direct        ⎦
                  [-no_console]
```

Arguments

- ▶ *message* **Required**

  The text of the message to be logged. A message cannot be longer than 256 characters, and it must be enclosed in apostrophes if it contains any spaces.

- ▶ `-module` *module_name*

  Specifies the module to receive the message. The default value is the current module. The `-module` and `-direct` arguments are mutually exclusive.

- ► `-direct` CYCLE **(Privileged)**

  Logs the message directly into OpenVOS's `syserr` buffer in the kernel, without going through the `Overseer`. The default is `no`, meaning the message goes through the `Overseer`. Use the `-direct` argument with the `log_syserr_message` command when you use it in the `module_start_up.cm` file. The `-module` and `-direct` arguments are mutually exclusive.

- ► `-no_console` CYCLE **(Privileged)**

  Specifies that messages sent by the `Overseer` and messages resulting from specifying `-direct` are not sent to the console. By default (the value `yes`), messages are logged to the console, whether or not a value is specified for `-module` or `-direct` is specified.

Examples

The following example shows how the `log_syserr_message` command is typically used in a `module_start_up.cm` file.

```
&if (command_status) ^= 0 &then log_syserr_message
    (string configure_modules: (message (command_status)))
-direct
```

Related Information

See the manual *OpenVOS System Administration: Starting Up and Shutting Down a Module or System* (R282) for additional information.

maint_request

Purpose

The `maint_request` command sends an RSN request from your site to the RSN HUB. Any user can invoke the `mail`, `add_call` and `update_call` requests. Only users defined as administrators in the `HUB.data` file can issue the remaining requests.

Display Form

```
-------------------------------- maint_request --------------------------------
request:       mail
args:
-priority:     standard
-defer_until:
-retry_limit:  5
-delete:       no
```

Command-Line Form

```
maint_request request
            [args]
            [-priority type]
            [-defer_until date/time]
            [-retry_limit number]
            [-delete]
```

Arguments

▶ *request* CYCLE **Required**

The request to be sent to the HUB; see the Explanation for detailed information:

`mail` (the default)	`unmask`	`comm_trace`
`update`	`demo_mode`	`truncate_queue`
`add_call`	`send_message`	`periodic_call`
`update_call`	`status`	`cancel_periodic_call`
`mask`	`stop_server`	

▶ `args`
  Arguments used with some of the requests.

▶ `-priority type` CYCLE
  Specifies how the request is to be scheduled. You can select one of the following values.

- `standard` (the default), which places the request in the queue at a standard priority, which provides the normal scheduling of the request
- `express`, which places the request in the queue at a high priority
- `nighttime`, which executes the request during the night when phone billing rates are likely to be lower (11:00 p.m. through 6:00 a.m.)
- `manual_pickup`, which holds the mail until the HUB calls. Use this value when there are problems with outgoing lines
- `low`, which places the request in the queue at a low priority

▶ `-defer_until date/time`
  Defers processing of the request until the specified date and time. If you omit this argument, the request is not deferred.

▶ `-retry_limit number`
  Limits the number of times the request is retried after common abort situations such as line hangup. If you omit this argument, the request is retried up to five times.

▶ `-delete` CYCLE
  Deletes local files once they have been successfully transferred to the HUB. The default is `no`, which means the files are not automatically deleted.

Explanation

Your site administrator can use the `maint_request` command to send RSN requests to the RSN HUB and to cause certain requests from the HUB to be ignored or recognized at the site.

You can send the following requests with the `maint_request` command.

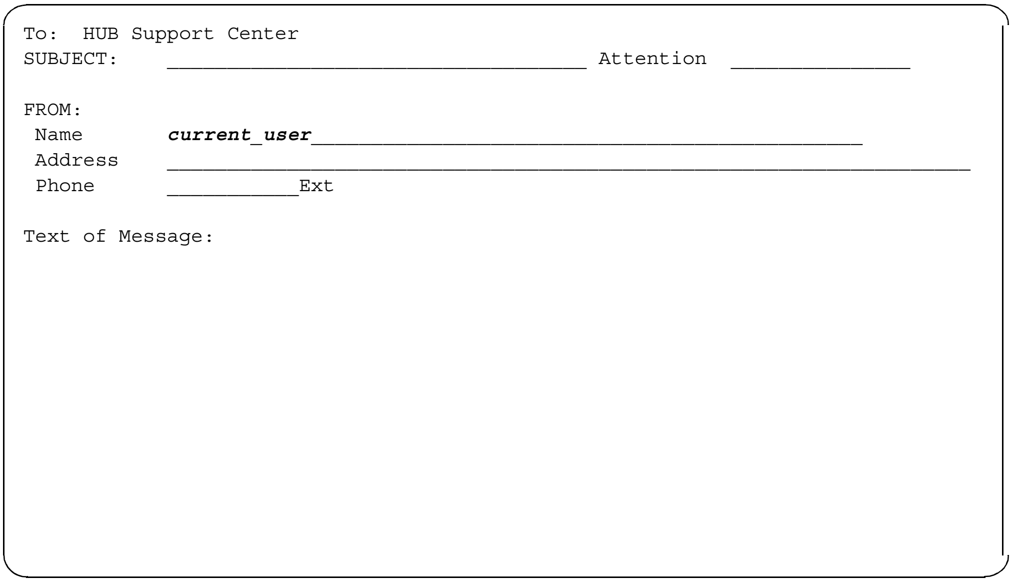
NOTE

The key sequences shown in the following screens are for the V105 terminal with the PC/+ 106 keyboard and the V105/V109 with the EPC keyboard. The OpenVOS Software Release Bulletin provides the corresponding key sequences for the V105 terminal with the ANSI keyboard.

▶ `mail`[*`path_name`*]

Transmits a piece of mail to the HUB. The *`path_name`* argument is the name of an existing file that is to be sent as mail. The file should be no larger than 4000 characters.

If you do not specify a path name, the RSN displays the following screen into which you can enter up to 10 lines of text.



```
To:  HUB Support Center
SUBJECT:     ___________________________________ Attention  _______________

FROM:
 Name        current_user______________________________________________
 Address     ___________________________________________________________________
 Phone       ___________Ext

Text of Message:
```

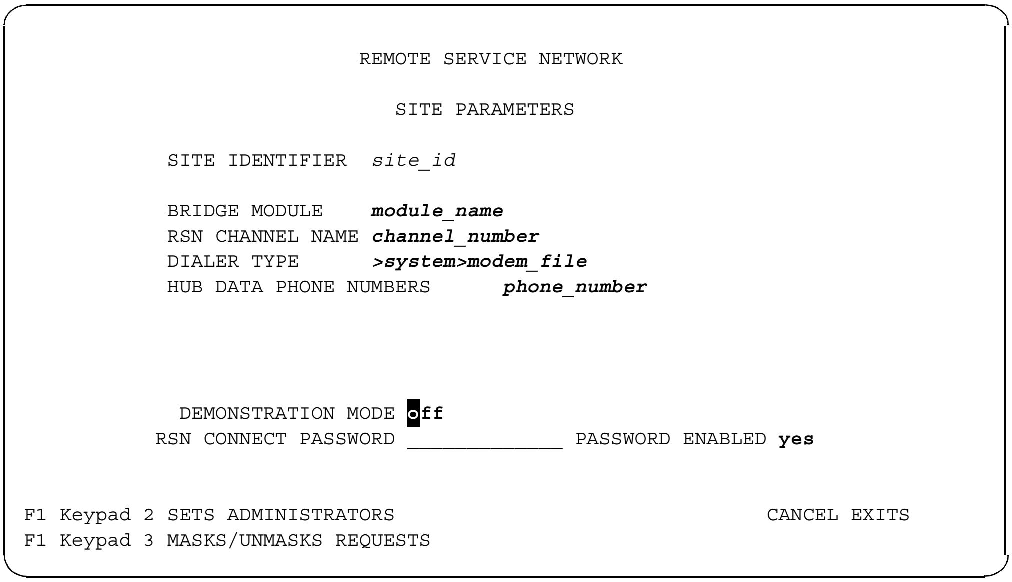
To submit the text as mail, press the Enter key on the keypad. The following message appears at the bottom of the screen.

```
Mail queued as site_id.u.(date).(time).
```

▶ `update`

Updates values contained in the file `HUB.data`. When you specify the `update` request, the `SITE PARAMETERS` screen appears. The `RSN ADMINISTRATORS` and `MASKED HUB REQUESTS` screens appear subsequently. To return to the `SITE PARAMETERS` screen after you have changed the screen display, press the F1 keypad 1 key combination.

`SITE PARAMETERS` Screen of the `update` Request



```
                         REMOTE SERVICE NETWORK

                            SITE PARAMETERS

             SITE IDENTIFIER  site_id

             BRIDGE MODULE    module_name
             RSN CHANNEL NAME channel_number
             DIALER TYPE      >system>modem_file
             HUB DATA PHONE NUMBERS      phone_number




              DEMONSTRATION MODE off
            RSN CONNECT PASSWORD _____________ PASSWORD ENABLED yes


 F1 Keypad 2 SETS ADMINISTRATORS                              CANCEL EXITS
 F1 Keypad 3 MASKS/UNMASKS REQUESTS
```

Most values are set automatically and should not be changed though you can modify the following:

- `DEMONSTRATION MODE`—Disables the automatic service call from the site to the HUB when a hardware error at your site is reported. The default is `off`, meaning that calls are automatically made.

  When demonstration mode is on, error messages are logged at the site to the file `(master_disk)>Overseer>h.`*`date`*`.`*`time`*. No call is placed to the HUB. This allows the site to perform demonstrations and tests that involve simulating failures.

- `RSN CONNECT PASSWORD`—Specifies the password that the HUB must supply whenever it dials your site. Inform the HUB by phone of the password entered into this field. If your system uses an RSN Internet console server, you must specify a password. The password is erased from the screen as soon as you press ←Enter; subsequently, the password is never displayed on the screen.

  NOTE

  You must coordinate any changes in the value of the `RSN CONNECT PASSWORD` field and, thus, the value of the *`rsn_connect_password`* argument of the `update_rsnip_site` command with the CAC or HUB because these two values must be identical. For more

]

information on this password, see the Explanation of the `update_rsnip_site` command description.

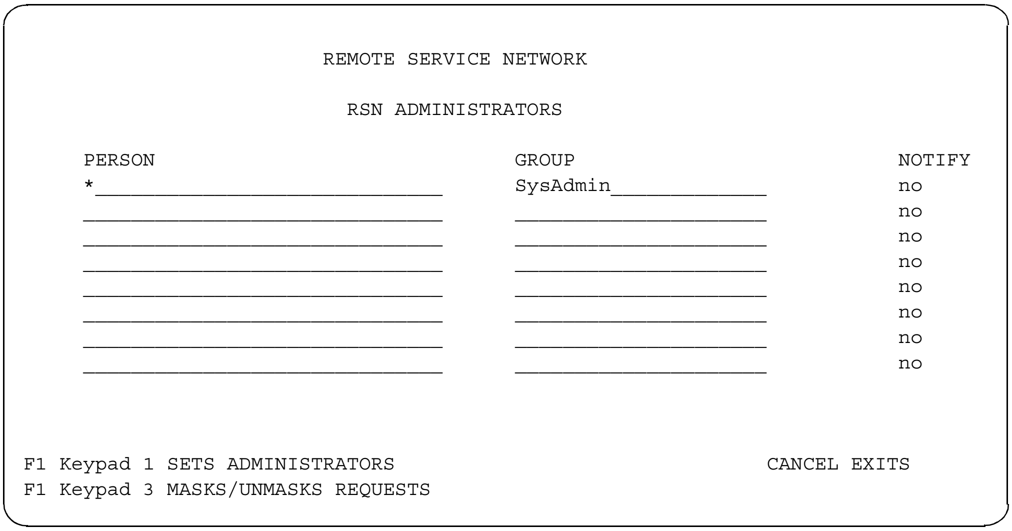
- `PASSWORD ENABLED`—Requires the HUB to supply a password (defined in the `RSN CONNECT PASSWORD` field) whenever it dials your site. If you enable this option, inform the HUB by phone.

  If your system uses an RSN Internet console server, you must set this value to `yes`. If the `PASSWORD ENABLED` field is set to `no`, the `maint_request` command fails and displays the following message:

  ```
  Could not update password on console server: RSN/IP
  requires a password.
  ```

RSN ADMINISTRATORS Screen of the update Request

To display the `RSN ADMINISTRATORS` screen, press the F1 keypad 2 key combination.



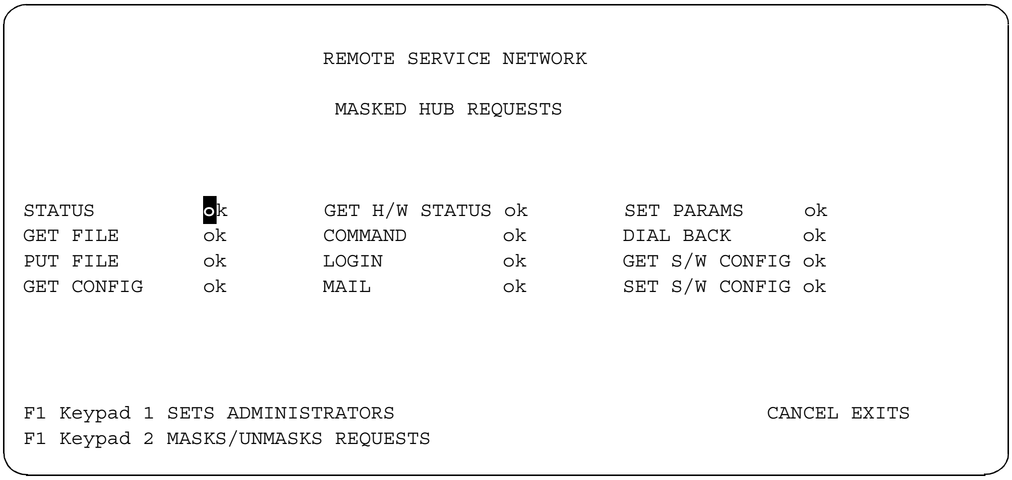
```
                          REMOTE SERVICE NETWORK

                            RSN ADMINISTRATORS

      PERSON                                   GROUP                           NOTIFY
      *_____________________________          SysAdmin_____________           no
      ______________________________          _____________________           no
      ______________________________          _____________________           no
      ______________________________          _____________________           no
      ______________________________          _____________________           no
      ______________________________          _____________________           no
      ______________________________          _____________________           no
      ______________________________          _____________________           no


 F1 Keypad 1 SETS ADMINISTRATORS                                   CANCEL EXITS
 F1 Keypad 3 MASKS/UNMASKS REQUESTS
```

The `RSN ADMINISTRATORS` screen displays the names of administrators of the RSN software at the site. You can supply up to eight names on this screen; if you want more than eight, create a group for two or more users, then enter the value `*.`*`group_name`* as an administrator name. A user can then function as an RSN administrator only when logged in as a member of the specified group.

This screen also displays the `NOTIFY` attribute for each administrator name. This attribute determines whether the specified administrator (or administrator group) is notified of files or mail received from the HUB and of interactions between the HUB and the site. Notification occurs on the 25th line of each administrator's terminal.

`MASKED HUB REQUESTS` Screen of the `update` Request

To display the `MASKED HUB REQUESTS` screen, press the F1 keypad 3 key combination.



```
                           REMOTE SERVICE NETWORK

                            MASKED HUB REQUESTS



STATUS          ok         GET H/W STATUS ok         SET PARAMS      ok
GET FILE        ok         COMMAND        ok         DIAL BACK       ok
PUT FILE        ok         LOGIN          ok         GET S/W CONFIG  ok
GET CONFIG      ok         MAIL           ok         SET S/W CONFIG  ok




F1 Keypad 1 SETS ADMINISTRATORS                                  CANCEL EXITS
F1 Keypad 2 MASKS/UNMASKS REQUESTS
```

The `MASKED HUB REQUESTS` screen displays a list of HUB requests and indicates whether each is currently designated as `masked` or `ok`. If a request is designated as `masked`, your site does not accept the request from the HUB. To change one of these values, press the ←Enter key to move the cursor to a specific request. Press the ← and → keys to change the value.

HUB personnel communicate with sites via a HUB command. This command has several requests. Many of the requests perform functions such as sending mail, files, and inquiries to sites. Other requests perform functions that affect only the HUB itself, such as notifying HUB staff of communications from other parts of the network.

At your site, an administrator can place the names of certain HUB requests on a list of requests that is stored in the `HUB.data` file. Requests on this list are masked requests, which means they are ignored by modules at the site. You place requests on the list with either the `mask` or `update` request of the `maint_request` command, and you remove requests with either the `unmask` or `update` request.

Your site administrator can mask and unmask the following HUB requests.

NOTES

1. Each HUB request is case-insensitive; you can type the request in either uppercase or in lowercase.
2. To change one of these values within the command line, issue the `maint_request mask` command with the request. For two-word requests, enclose the words in apostrophes (`' '`) and include an underbar (or underline) character (_) between the two words, as in the following `GET FILE` (`get_file`) example:
   `maint_request mask 'get_file'`

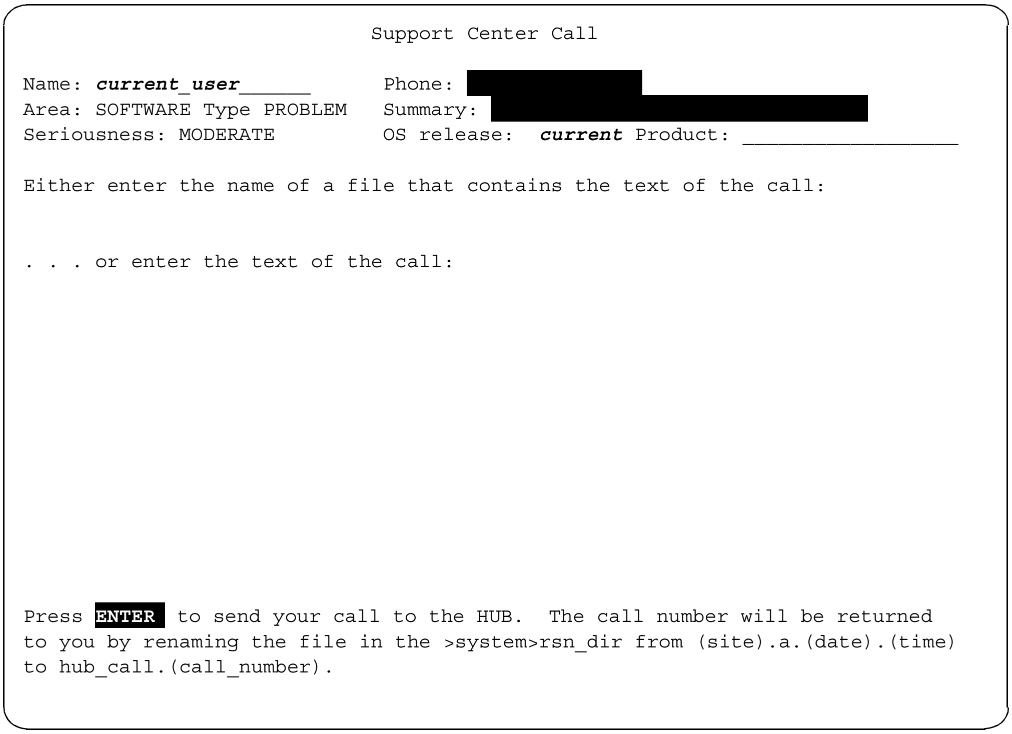
- `STATUS`—Displays the following status information about your site.
  - whether each module is online or offline
  - the time at which each online module was last booted
  - the software release in use on each module
  - the requests that are masked at the site
  - the administrators for the site
- `GET FILE`—Transmits a file from your site to the HUB.
- `PUT FILE`—Transmits a file from the HUB to your site.
- `GET H/W STATUS`—Displays a summary of the site's current hardware status. It includes the following information for the specified module on the site.
  - the boards that are configured, present, and broken
  - the disks that are configured, present, and broken
- `COMMAND`—The MaintBridgeServer process on the specified module at your site creates a process that executes the command specified as an argument to the `command` request. The command's output is placed in a file that is returned to the HUB.
- `LOGIN`—Allows a person at the HUB to log in to your site, using the channel assigned to the MaintBridgeServer process.
- `MAIL`—Sends mail to your site.
- `SET PARAMS`—Specifies the threshold values that the HUB uses to determine when it should be notified of hardware errors.
- `DIAL BACK`—Permits loopback tests of communications between the HUB and your site. With `DIAL BACK`, the HUB asks the site to dial the HUB and return the request message.
- `GET S/W CONFIG`—Displays the list of software packages installed at your site.

- `SET S/W CONFIG`—Modifies the list of software packages installed at your site.

▶ `add_call` [`path_name`]

Displays a screen you can use to submit a call to the HUB. You can either type the call information directly into the screen or type in the path name of a file that contains the information. When you press the ←Enter key, the information is transmitted to the HUB's call database and is assigned a call number, which is communicated to the site via RSN mail. The system makes five attempts to transmit the file. If, for any reason, the transmission fails after five attempts, resubmit the file.

The `add_call` request displays the following screen.



```
                              Support Center Call

Name: current_user______      Phone:
Area: SOFTWARE Type PROBLEM   Summary:
Seriousness: MODERATE         OS release:  current Product: __________________

Either enter the name of a file that contains the text of the call:


. . . or enter the text of the call:













Press ENTER  to send your call to the HUB.  The call number will be returned
to you by renaming the file in the >system>rsn_dir from (site).a.(date).(time)
to hub_call.(call_number).
```

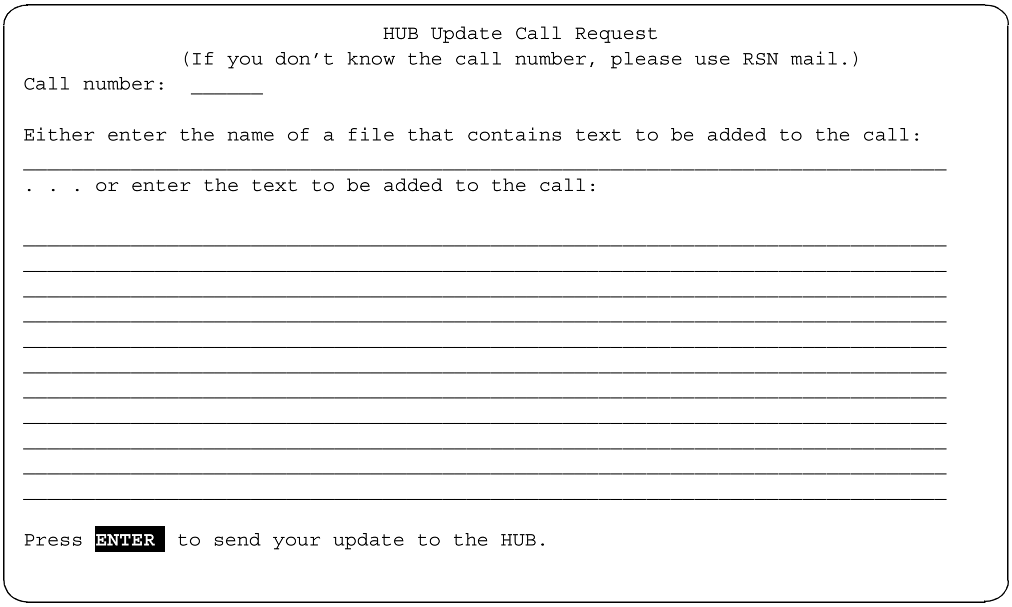
See the description of the `add_call` request of the `site_call_system` command later in this chapter for a discussion of calls and the fields in call screens.

▶ `update_call` [`path_name`]

Displays a screen you can use to update a call previously submitted to the HUB. It appends the `.`*`date`*`.`*`time`* stamp to the file name in order to identify the call. The file format is *`site_id`*`.u.`*`(date)`*`.`*`(time)`*, which corresponds to the call number format. For more information about the call number format, see the `site_call_system` command description later in this chapter.

All calls that have been assigned call numbers are listed as files in the directory `(master_disk)>system>rsn_dir`.

The `update_call` request displays the following screen.



```
                              HUB Update Call Request
             (If you don't know the call number, please use RSN mail.)
Call number:  ______

Either enter the name of a file that contains text to be added to the call:
_____________________________________________________________________________
. . . or enter the text to be added to the call:

_____________________________________________________________________________
_____________________________________________________________________________
_____________________________________________________________________________
_____________________________________________________________________________
_____________________________________________________________________________
_____________________________________________________________________________
_____________________________________________________________________________
_____________________________________________________________________________
_____________________________________________________________________________
_____________________________________________________________________________
_____________________________________________________________________________

Press ENTER to send your update to the HUB.
```

- ▶ `mask` *`request_names`*

  Names HUB requests that the MaintBridgeServer process at the site is not to accept. The names and meanings of requests that you can mask are listed after the `MASKED HUB REQUESTS` screen, documented earlier in this command description.

  When you invoke `mask` in the display form, enter the request name(s) to unmask after the *`args`* argument. If you specify more than one request name, enclose the list in apostrophes. (`'`*`request request`*`'`).

- ▶ `unmask` *`request_names`*

  Names HUB requests that have been masked but are again executable.

  When you invoke `unmask` in the display form, enter the request name(s) to unmask after the *`args`* argument. If you specify more than one request name, enclose the list in apostrophes.

- `demo_mode`
  Turns demonstration mode off or on, thereby controlling whether messages are sent to the HUB. If you specify `demo_mode` with the value `on` for *`args`*, `maint_request` turns on demonstration mode and disables the sending of messages to the HUB. If you specify `demo_mode` with the value `off` for *`args`*, `maint_request` turns off demonstration mode and enables the sending of messages to the HUB. Be sure to specify `demo_mode on` when you begin conducting tests or demonstrations; specify `demo_mode off` when you are done the testing or demonstrations.

- `send_message '`*`message`*`'`
  Displays the message on the 25th line of the terminal of the user currently logged in to your system over the RSN line. This request performs the same function as the `send_message` command. For a description of the `send_message` command, see the *OpenVOS Commands Reference Manual* (R098).

- `status`
  Verifies that the MaintBridgeServer process is running.

- `stop_server`
  Stops the MaintBridgeServer process and ends communication with the HUB.

- `comm_trace`
  A debugging tool that should be used only after consultation with the Stratus Customer Assistance Center (CAC) or your authorized Stratus service representative.

- `truncate_queue`
  Truncates the file `HUB.comm_MQ`, discarding messages that were waiting in the queue to be sent to the HUB.

- `periodic_call`[*`frequency`*]
  Places in the queue a simple message to be sent to the HUB after a specified number of hours, as specified in the *`args`* argument.

  The purpose of `periodic_call` is to allow sites to automatically call the HUB at a given frequency. Frequency is the number of hours between periodic calls, specified by a number in the *`args`* field when you make the `periodic_call` request. If the frequency is less than or equal to zero, only one call is placed.

  This is useful for sites that do not normally allow the HUB to dial in.

  The contents of the periodic call message to the HUB is simply the date and time of the next expected periodic call. This provides sufficient information for the HUB to determine whether the site is having communication problems. For example, if an expected call does not arrive at the specified time, HUB personnel can investigate the situation.

If you request it, a flag can be set in the site database to disable HUB logins and file transfers. In this situation, a file transfer request is queued and remains in the queue until you initiate contact. Then issuing a `periodic_call` request with `line_swap` enabled allows you to pick up any mail queued for your site by HUB personnel.

- `cancel_periodic_call`
  Removes from the queue any periodic messages waiting to be sent to the HUB. (Under most circumstances only one periodic call request will be in the queue at one time.)

Examples

The following command sends the file `log_info` as mail to the HUB.

```
maint_request mail log_info
```

The following command displays screens for updating values in the file `HUB.data`.

```
maint_request update
```

The following command causes the requests `mail` and `login` not to be accepted at your site.

```
maint_request mask 'mail login'
```

The following command causes the `login` request to be accepted at your site.

```
maint_request unmask login
```

Related Information

See Chapter 6, "The Remote Service Network," and the descriptions of the commands `rsn_mail` and `site_call_system` later in this chapter.

make_message_file

Purpose

The `make_message_file` command processes the file `error_codes.table` to create the file `error_codes.text`.

Display Form

```
----------------------------- make_message_file -----------------------------
input_path:  error_codes.table
output_path: current_directory>error_codes.text
```

Command-Line Form

```
make_message_file [input_path]
            [output_path]
```

Arguments

- ▶ *input_path*

  The path name of the `error_codes.table` file. The default value is the current directory, which must contain a `error_codes.table` file. Execute this command only from the directory `(master_disk)>system>error`, where an `error_codes.table` file is stored. If you specify a value for this argument, do not specify an extended name.

  If the `error_codes.table` file is missing from the `(master_disk)>system>error` directory, first run `update_codes` to create `error_codes.table`.

- ▶ *output_path*

  The path name of the output file (named `error_codes.text`) created by the command. If you do not specify a value, OpenVOS places the file in the current directory. If you specify a value for this argument, do not specify an extended name.

Explanation

The make_message_file command creates a compressed version of the OpenVOS error message file created with the create_table command.

This command expects to find sequentially numbered error codes when processing error_codes.table. If it finds that error codes are missing, it continues execution, but returns warning messages such as the following:

```
make_message_file:  There is no message code 222.
make_message_file:  There are no message codes from 6 to 222.
```

One of the functions of the update_codes command is to create the file error_codes.text. In most cases, you will use update_codes and therefore not need make_message_file.

Related Information

See Chapter 9, "Customizing Status Codes and Messages," and the description of the command update_codes later in this chapter, as well as the description of the command create_table in the manual *OpenVOS System Administration: Configuring a System* (R287).

monitor_rsn_servers

Purpose

This is a system process command. It is typically used within `mddmon_start_up.cm` files.

The `monitor_rsn_servers` command checks if the Remote Service Network processes (MaintServer and MaintBridgeServer) are running. This command is typically issued within a `start_process` command line in the `mddmon_start_up.cm` file.

Display Form

```
--------------------------- monitor_rsn_servers ---------------------------
No arguments required.  Press ENTER to continue. █
```

Command-Line Form

```
monitor_rsn_servers
```

Explanation

The `monitor_rsn_servers` command checks once each hour if the Remote Service Network processes (MaintServer and MaintBridgeServer) are running. If one or both of the processes are not running, the command logs a warning message, once per day, in the `syserr_log.(date)` file (in the `(master_disk)>system` directory). The error message has the following form:

```
HH:MM:SS User_Name (monitor_rsn_servers) : Remote Service Network:
    Hardware Calls Will Not Be Escalated!
    The Maintenance Server is not running.

HH:MM:SS User_Name (monitor_rsn_servers) : Please Restart the RSN.
```

The `monitor_rsn_servers` command is typically specified in the command line of a `start_process` command in a `mddmon_start_up.cm` file. To use the `monitor_rsn_servers` command, edit the `mddmon_start_up.cm` file, adding the following lines before the `start_rsn` command:

```
&
&  Use monitor_rsn_servers.pm to warn if one of the
&    Remote Service Network processes stop.
&
&  The warning message will be in (master_disk)>system>syserr_log.(date)
&
&  More information about the RSN processes
&   (MaintServer and MaintBridgeServer)
&   can be found at
&   http://stratadoc.stratus.com/
&
&if (exists monitor_rsn_servers.out)
&then !rename monitor_rsn_servers.out monitor_rsn_servers.out.old -delete
&
!create_file  monitor_rsn_servers.out
!set_implicit_locking  monitor_rsn_servers.out on
!start_process monitor_rsn_servers.pm
&
```

When it starts, the `monitor_rsn_servers` process sleeps 15 minutes before checking RSN processes for the first time, which allows time for the RSN processes to start and establish connections.

The `monitor_rsn_servers` process logs messages that are informational, only. If necessary, a system administrator must take corrective action to restart the MaintServer process and/or the MaintBridgeServer process.

Related Information

In this chapter, see descriptions of the `start_rsn` and `maint_request` commands. See also Chapter 6, “The Remote Service Network.”

network_watchdog **Privileged**

Purpose

This is a system process command. It is typically used within `module_start_up.cm` files.

The `network_watchdog` command causes the executing process to operate as a network watchdog.

Display Form

```
------------------------------ network_watchdog -----------------------------
num_minutes: 2
num_seconds: 0
-syserr:     no
```

Command-Line Form

```
network_watchdog [num_minutes num_seconds]
            [-syserr]
```

Arguments

▶ *num_minutes*
The time interval, in minutes, between successive network watchdog cleanup operations. The default value is 1. For large StrataNET configurations, use a longer interval to reduce overhead.

▶ *num_seconds*
The time interval, in seconds, between successive network watchdog cleanup operations. The default value is 0. To fine-tune your network, specify a value of `0` for the *num_minutes* argument, and specify a value of `1` or greater for this *num_seconds* argument.

▶ `-syserr` CYCLE
Logs messages generated by this network watchdog process to the system error log. The default value is `no`, which suppresses the logging of messages.

Explanation

Normally, the `network_watchdog` command is specified in the command line of a `start_process` command in a `module_start_up.cm` file. The `network_watchdog` command usually invokes the `-syserr` argument. This command is **never** executed by more than one process simultaneously.

Examples

The following command, shown as it might appear in a `module_start_up.cm` file, specifies that network watchdog cleanup operations occur every 5 minutes.

```
start_process 'network_watchdog 5 -syserr' -privileged
-priority 9
```

The following command, shown as it might appear in a `module_start_up.cm` file, specifies that network watchdog cleanup operations occur every 15 seconds.

```
start_process 'network_watchdog 0 15 -syserr' -privileged
-priority 9
```

Related Information

See also the `module_start_up.cm` file shipped with the installation software. It is stored in the `(master_disk)>system>release_dir` directory.

notify_hardware_error

Purpose

The `notify_hardware_error` command either starts or stops the notification of hardware errors to a specified terminal. When notification is enabled for a terminal, OpenVOS displays a message on that terminal whenever an entry is written to the `hardware_log.(date)` file of a module.

Display Form

```
---------------------------- notify_hardware_error -------------------------
terminal: █
-module:
-off:     no
```

Command-Line Form

```
notify_hardware_error [terminal]
            [-module module_name]
            [-off]
```

Arguments

- *terminal*

  The path name of a terminal to which notification is to be started or stopped. The default is the terminal of the current process. The path name of a terminal must be preceded by the `#` symbol.

  If you do not specify a value for this argument, notification to the terminal running the current process stops when that process logs out. If you do specify a value, notification to the named terminal continues, even after the current process logs out, until you explicitly stop notification with the `-off` argument.

- `-module` *module_name*

  Specifies the name of the module whose hardware errors are to be signaled, or are no longer to be signaled, to the specified terminal. The default is the current module.

▶ `-off` CYCLE

Disables notification to the specified terminal. The default is `no`, which enables notification.

Explanation

Notification consists of a message, displayed on the status line of the specified terminal, indicating that an entry was written to the file `hardware_log.(date)`. The message includes the device type (disk, communications device, and so on) causing the error, but it does not display the full text of the error. The minimum time between notifications is two minutes. If no device had an error in the preceding two minutes, no message is sent.

A terminal receives notifications until the process controlling that terminal is either logged out (if no value was specified for the *`terminal`* argument) or until the terminal that was explicitly named in the *`terminal`* argument is named in a subsequent `notify_hardware_error` command with the `-off` argument.

Examples

The following command starts notifying terminal `#term.3.12` of hardware errors on module `m3`.

```
notify_hardware_error #term.3.12 -module m3
```

Related Information

See the manual *OpenVOS System Administration: Starting Up and Shutting Down a Module or System* (R282).

overseer *Privileged*

Purpose

This is a system process command. It is used **only** within `module_start_up.cm` files.

The `overseer` command starts the Overseer process, which performs numerous functions such as starting processes for all login terminals, creating the files `syserr_log.(date)` and `hardware_log.(date)`, monitoring batch queues with the BatchOverseer process, enabling emergency shutdown, and installing files specified in the `broadcast_file` command. This command also enables you to indicate whether the Overseer should start disk recovery during Overseer initialization. In addition, this command can send copies of log-file messages to remote servers.

Display Form

```
---------------------------------- overseer -------------------------------
-start_disk_recovery:              yes
-prelogin_out_of_service_time:  30
-max_shutdown_minutes:          65
-min_shutdown_writes:           100
-syslog_host:
-syslog_tag:                    no
```

Command-Line Form

```
overseer [-no_start_disk_recovery]
         [-prelogin_out_of_service_time number]
         [-max_shutdown_minutes number]
         [-min_shutdown_writes number]
         [-syslog_host host_name]
         [-syslog_tag]
```

Arguments

▶ `-no_start_disk_recovery` CYCLE

Suppresses the start of disk recovery. The default is `yes`, meaning disk recovery starts during Overseer initialization.

▶ `-prelogin_out_of_service_time` *`number`*

Specifies the amount of time, in seconds, that a channel is taken out of service due to prelogin failures. The default is `30`.

▶ `-max_shutdown_minutes` *`number`*

The maximum number of minutes allowed for user processes to terminate during shutdown. The minimum value is `5`, the maximum value is `120`, and the default value is `65`.

▶ `-min_shutdown_writes` *`number`*

The minimum number of disk writes needed per minute during the user process termination phase of a shutdown, in order for the shutdown time to be extended. Extending the shutdown time prevents the shutdown from ending too soon when user processes have many modified pages that need to be written out to disk, and the disks are still operational.

The shutdown is extended in three-minute increments, up to the amount of time specified in the `-max_shutdown_minutes` argument.

▶ `-syslog_host` *`host_name`*

Specifies the name of the host that will receive the log-file messages. The *`host_name`* value can be either an IP address or a name that is resolvable by the DNS.

▶ `-syslog_tag` CYCLE

Prefixes one of the following tags to the log-file messages to indicate from which log file the message originated. Note that the colon is part of the tag.

- `VOS-SYSERR:`. The message originated from the `syserr_log.(date)` file.
- `VOS-HARDWARE:`. The message originated from the `hardware_log.(date)` file.
- `VOS-SEL:`. The message originated from the `sel_log.(date)` file.
- `VOS-SECURITY:`. The message originated from the `security_log.(date)` file.

By default (`no`), the command does not prefix a tag to a log-file message.

Explanation

The argument `-start_disk_recovery` notifies the Overseer to start disk recovery during overseer initialization.

If you use `-no_start_disk_recovery`, disk recovery is not started. In this case, you can use the command `start_disk_recovery`, either manually or in the `module_start_up.cm` file, to start disk recovery at a later time.

The Overseer process initiates a shutdown and tracks its progress, after you issue the `shutdown` command. An important part of an orderly shutdown is terminating all user processes, allowing them sufficient time to close out of all of their files. A significant amount of time may be required for all user processes to close out of all files, if the processes need to write out many modified blocks or large, modified shared virtual-memory regions.

Ending shutdown before the processes have finished writing their data could result in lost data. The Overseer process can track progress during shutdown and prolong shutdown in three-minute increments, up to the maximum specified by the `-max_shutdown_minutes` argument, as long as discernible progress is being made. To avoid a hang during shutdown, specify values for the `-max_shutdown_minutes` and `-min_shutdown_writes` arguments, in order to adjust thresholds that affect the maximum time a shutdown can be extended while waiting for user processes to terminate.

The Overseer can use the `syslog` protocol to send copies of log-file messages to remote servers. The command can send messages from system error, hardware, system event (SEL), and security log files remotely. You can use this feature with any `syslog` daemon that is compatible with the standard `rsyslog` implementation.

Examples

The following example shows the `overseer` command as it might appear in a `module_start_up.cm` file if you choose to delay disk recovery.

```
start_process  'overseer -no_start_disk_recovery' -priority 8
   -output_path overseer.out -process_name TheOverseer -privileged
```

In the following example, the `overseer` command changes the maximum time allowed for user processes to terminate during shutdown from 65 minutes to 90 minutes, and it changes the minimum number of disk writes needed per minute, during the user process termination phase of a shutdown, from 100 to 150. This command, then, extends shutdown for up to 90 minutes as long as user processes continue to write 150 blocks or more per minutes.

```
start_process 'overseer -max_shutdown_minutes 90 -min_shutdown_writes &+
   150' -priority 8 -process_name TheOverseer -privileged
```

Related Information

See the manual *OpenVOS System Administration: Starting Up and Shutting Down a Module or System* (R282) for a description of the `shutdown` command. See also the `module_start_up.cm` file shipped with the installation software. This file is stored in the directory `(master_disk)>system>release_dir`. For a description of the `broadcast_file` command, see the *OpenVOS Commands Reference Manual* (R098).

process_broadcast_queue

Purpose

The `process_broadcast_queue` command manages the requests in the file `broadcast_queue.message_queue`. It is used by the FileBroadcaster process to broadcast files. The FileBroadcaster process is started by the Overseer when a user invokes the `broadcast_file` command.

Display Form

```
--------------------------- process_broadcast_queue ---------------------------
No arguments required.  Press ENTER to continue. █
```

Command-Line Form

```
process_broadcast_queue
```

Explanation

The `process_broadcast_queue` command copies a file to a target directory in all modules of a system. The target directory must be a subdirectory of the master disk directory on each module.

A request in the `broadcast_queue.message_queue` file remains there until all modules on that module's system (except the excluded modules, if any) have received a copy of the file named in the request. In this way, files are eventually copied to all modules in a system, even if some modules are not in operation at the time a request is placed in the `broadcast_queue.message_queue` file.

If a file, with the same name as the file being copied, exists in the target directory, `process_broadcast_queue` normally overwrites that existing file. However, if the existing file is open, `process_broadcast_queue` renames it with a unique name that includes the suffix `.old`. When the `.old` file is no longer in use, it is deleted.

The command `process_broadcast_queue` logs the following information in the `(master_disk)>system>broadcast_log.(date)` file.

- the time that processing began on each request
- the source file for each request
- the destination of each file that was broadcast

The Overseer generates output when it creates a FileBroadcaster process to execute `process_broadcast_queue`. The output is written to the file `(master_disk)>system>file_broadcaster.log`.

Examples

```
97-11-02 13:21:36 EDT
%s1#d01>Sales>Smith>install>commands>process_broadcast_queue
.pm by Smith.SysAdmin
%s1#d02>system>command_library>process_broadcast_queue.pm
%s1#d018>system>command_library>process_broadcast_queue.pm
```

Related Information

See the commands `broadcast_file`, `cancel_broadcast_requests`, and `list_broadcast_requests`, documented in the manual *OpenVOS System Administration: Configuring a System* (R287).

rsn_mail

Purpose

The `rsn_mail` command allows users to send and receive mail over the Remote Service Network and to read, print, and delete that mail. Only users who have write access to the directory `(master_disk)>system>rsn_dir` can use this command.

Display Form

```
------------------------------------ rsn_mail --------------------------------
No arguments required.  Press ENTER to continue. █
```

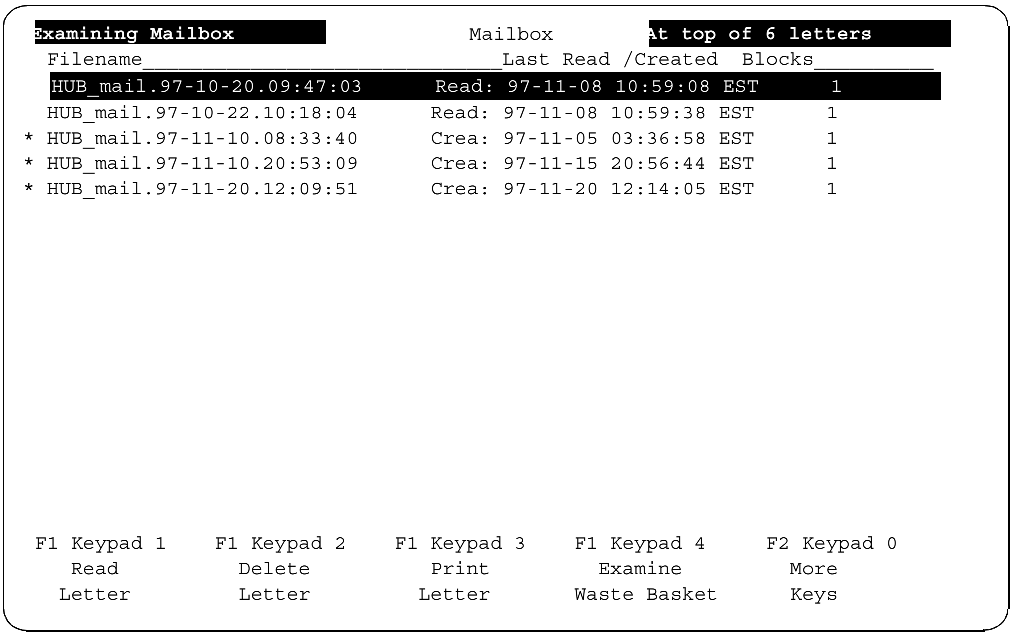
Command-Line Form

```
rsn_mail
```

Explanation

The `rsn_mail` command displays a mailbox containing mail sent between the module and the HUB. Function keys permit users to send, read, print, and delete mail.

The `Examining Mailbox` screen is the first screen displayed.



```
Examining Mailbox                         Mailbox          At top of 6 letters
  Filename______________________________Last Read /Created  Blocks__________
  HUB_mail.97-10-20.09:47:03        Read: 97-11-08 10:59:08 EST       1
  HUB_mail.97-10-22.10:18:04        Read: 97-11-08 10:59:38 EST       1
* HUB_mail.97-11-10.08:33:40        Crea: 97-11-05 03:36:58 EST       1
* HUB_mail.97-11-10.20:53:09        Crea: 97-11-15 20:56:44 EST       1
* HUB_mail.97-11-20.12:09:51        Crea: 97-11-20 12:14:05 EST       1









F1 Keypad 1    F1 Keypad 2    F1 Keypad 3    F1 Keypad 4     F2 Keypad 0
   Read          Delete          Print         Examine          More
  Letter         Letter         Letter       Waste Basket       Keys
```

The column headed by `Filename` lists the name of the file, and the date and time it was mailed from the HUB. The `*` labels a piece of mail as being unread. The column headed by `Last Read / Created` lists when the mail was last read, or when it was received by the mailbox. The column headed by `Blocks` lists the size, in blocks, of the mail.

Within each mail list, highlighting identifies the letter to be selected. To select a different letter, press the ↑ or ↓ key to move the highlighting to that letter on the list.

This section briefly explains all `rsn_mail` requests.

NOTES

1. The key sequences shown in this section are for the V105 terminal with the PC/+ 106 keyboard and the V105/V109 with the EPC keyboard. The OpenVOS Software Release Bulletin provides the corresponding key sequences for the V105 terminal with an ANSI keyboard.
2. The `CANCEL` function is active on each screen, even when it does not appear on the list of active keys.

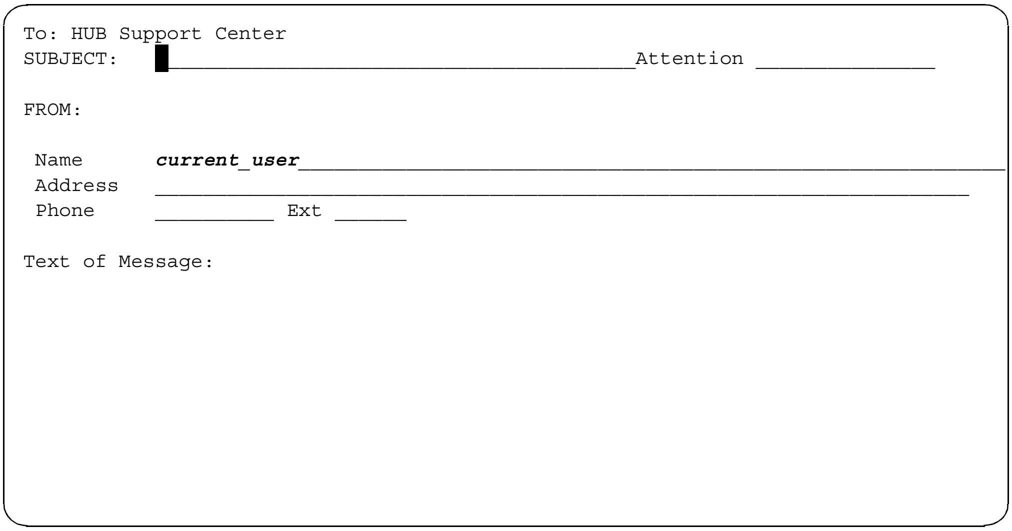
`Read Letter`
: Displays the highlighted letter.

`Delete Letter`
: Deletes the highlighted letter. The letter is moved to the wastebasket for storage. Letters are protected for at least two days in the wastebasket but will be permanently deleted within two weeks, depending on the scheduling of the `rsn_mail_janitor`.

`Print Letter`
: Enters a print request for the letter on the print queue. After you enter this command, the program prompts you for a queue name and a module name. The default is the `standard` queue on the current module. Pressing the * key on the keypad cancels the print request. Pressing the ←Enter key queues the letter.

`Examine Waste Basket`
: Displays the `Examining Waste Basket` screen, which lists recently deleted mail. This screen has the same columns as the `Examining Mailbox` screen.

`Send RSN Mail`
: Transmits a piece of mail to the HUB. The request displays a screen in which you can enter up to 10 lines of text; this screen is displayed below. To send the text as mail to the HUB, press the ←Enter key.



```
To: HUB Support Center
SUBJECT:  █___________________________________Attention ______________

FROM:

 Name      current_user_______________________________________________________
 Address   _______________________________________________________________
 Phone     _________  Ext ______

Text of Message:
```

The next two requests have parallel functions in the `maint_request` command. See the `maint_request` command for information about adding and updating RSN calls.

`Add an RSN Call`
Displays a form you can use to submit a call to the HUB. You either type the call information directly into the form, or type in the path name of a file that contains the information. When you press the Enter key on the keypad, the information is sent to the HUB's call database and is assigned a call number, which is communicated to the site via RSN mail.

`Update an RSN Call`
Displays a form you can use to update a call previously submitted to the HUB. To use `Update an RSN Call`, you must know the HUB call number, which has the format specified by the HUB servicing your system. For more information about the call number format, see the `site_call_system` command description later in this chapter.

All calls that have been assigned call numbers are listed as files in the directory `(master_disk)>system>rsn_dir`.

`Scroll Window Up`
Moves the contents of the window down one screen if letters are listed off the screen.

`Scroll Window Down`
Moves the contents of the window up one screen if letters are listed off the screen.

`Top of List`
Moves to the top of the list of letters.

`Bottom of List`
Moves to the bottom of the list of letters.

`Exit Program`
Found in all menus; exits from the program and returns to command level. In `Examine Mailbox`, the * key on the keypad behaves the same as the `Exit Program` request.

`More Keys`
Replaces the current set of available requests with a different set. Repeating this request cycles back to the original set of requests.

Related Information

See Chapter 6, "The Remote Service Network." See also the descriptions of the commands `maint_request`, `start_rsn`, `rsn_mail_janitor`, and `site_call_system` in this chapter.

rsn_mail_janitor

Purpose

The `rsn_mail_janitor` command automatically deletes mail in the RSN wastebasket for a module every two weeks.

Display Form

```
----------------------------- rsn_mail_janitor -------------------------------
rsn_waste_basket_dir: current_rsn_waste_basket_dir
```

Command-Line Form

`rsn_mail_janitor` *`rsn_waste_basket_dir`*

Arguments

▶ *`rsn_waste_basket_dir`*
The path name of the wastebasket directory that is to be serviced. If you omit this argument, `rsn_mail_janitor` deletes mail in the directory `(master_disk)>system>rsn_dir>waste_basket` on the current module.

Explanation

Issuing this command deletes from the RSN mail wastebasket on a module all letters that have been there longer than two days, then places in the `normal` batch queue a batch request that repeats this operation every two weeks.

You need to reissue the `rsn_mail_janitor` command only if the `normal` batch queue is deleted and re-created, or if the batch request the command issued is canceled (with the `cancel_batch_request` command). Stopping the `normal` batch queue with the `batch_admin` command may temporarily prevent the batch request issued by the `rsn_mail_janitor` command from running until that batch queue is started again, but this does not require reissuing the `rsn_mail_janitor` command.

Related Information

See Chapter 6, ‘‘The Remote Service Network,’’ and the descriptions of the `start_rsn`, `rsn_mail`, and `site_call_system` commands in this chapter.

`set_date_time` ***Privileged***

Purpose

You use the `set_date_time` command to perform the following tasks.

- set the date and time on the current module (either directly or from the module's calendar/clock)
- change the setting of the module's calendar/clock

You should rarely, if ever, need to issue the `set_date_time` command.

Display Form

```
-------------------------------- set_date_time ------------------------------
date/time:              █
-from_calendar_clock: no
```

Command-Line Form

```
set_date_time [date/time]

             [-from_calendar_clock]
```

Arguments

▶ *date/time*

The date and time to which the current module and/or module calendar/clock are to be set.

The date component describes both the year/month/day sequence and the field delimiters. The following table describes the valid symbols.

Symbols	Description
Year symbols	*YYYY* or *yyyy* represents four-digit years; *YY* or *yy* represents two-digit years
Month symbols	*MM* or *mm*
Day symbols	*DD* or *dd*
Delimiters	hyphen (`-`), slant (`/`), or period (`.`)

For example, the standard date format is described as *YY-MM-DD*, which formats December 21, 2010 as `10-12-21`. The format *DD.MM.YYYY* formats the same date as `21.12.2010`. The default is the standard date format.

The time component describes both the hour/minute/second sequence and the field delimiter. The following table describes the valid symbols.

Symbols	Description
Hour symbols	*HH* or *hh*
Minute symbols	*MM* or *mm*
Second symbols	*SS* or *ss*
Delimiter	colon (`:`)

For example, the standard time format is described as *HH-MM-SS*, which formats 1:15 p.m. as `13:15:00`.

The date and time components can also be a character string in any format accepted by the `(date_time)` and `(iso_date_time)` command functions. In this case, the string must be enclosed in apostrophes.

▶ `-from_calendar_clock` CYCLE

Sets the date and time of the module from the calendar/clock. The default is `no`.

If you specify this argument, you cannot specify the *date/time* argument.

Explanation

A module's default date and time and/or calendar/clock setting represent the number of seconds that have elapsed since the beginning of the OpenVOS epoch. Beginning on January 1, 1980, 00:00:00 Greenwich Mean Time (GMT), the *OpenVOS epoch* defines the period of time in which OpenVOS will operate predictably (that is, as described in the Stratus documentation set). The OpenVOS epoch ends on December 31, 2048, 23:59:59 GMT.

A module's date and time and/or calendar/clock setting may be restricted by the local time zone. OpenVOS supports many local time zones, some of which vary from GMT by as much as 13 hours. As a result of time-zone differences, the first full day that OpenVOS could operate predictably in any time zone was January 2, 1980. Similarly, the last full day that OpenVOS can operate predictably in any time zone is December 30, 2048. For information about changing a module's time zone, see the description of the `set_default_time_zone` command.

You cannot set a module's date and time and/or calendar/clock setting to a date and time that occurs before or after the OpenVOS epoch. You can, however, set the date and time and/or calendar/clock setting to a past or future date that occurs within the OpenVOS epoch. If you set a new date and time that is two weeks or more from the current date and time on the module, the `set_date_time` command, in most cases, prompts you to confirm that the new date and time are correct, as illustrated in Examples. The `set_date_time` command prompts you to confirm if you want to specify the new date and time directly or obtain it from the calendar/clock.

NOTE

If the calendar/clock setting falls on January 1, 1980 at 00:00:00 or within two weeks after that date/time, the `set_date_time` command accepts the date and time you specify without prompting you for confirmation.

Resetting a module's date and time and/or calendar/clock may cause OpenVOS, and any applications running on OpenVOS, to behave unpredictably. If you incorrectly set a module's date and time and/or calendar/clock, you may experience operating-system and/or application errors relating to date/time inconsistencies. For example, if you set a module's calendar/clock to a date and time in the future, the mail handler may delete mail that would ordinarily be retained for the current date and time. Also, passwords may expire if you set a module's calendar/clock to a date and time in the future.

Examples

The following command sets the date and time on the current module directly.

```
set_date_time 2010-11-01_11:05:00
```

The following command sets the date and time on the current module to the beginning of the OpenVOS epoch.

```
set_date_time 1980-01-01_00:00:00
```

The following command sets the current module's date and time from the calendar/clock.

```
set_date_time -from_calendar_clock
```

The following command sets the time on the current module directly.

```
set_date_time 14:46:00
```

The following command illustrates the prompt received when the new date is two weeks or more from the current date. (Assume that the current date is 10-11-01, where the year is 2010 and the month is November.)

```
set_date_time 10-11-21
A significant difference in dates exists.  Are you sure the
new date, 10-11-21 00:00:00 EST, is correct?  (yes, no)
```

Related Information

For information about the date/time formats, see the manual *OpenVOS System Administration: Configuring a System* (R287). For information about the date/time command functions, see the *OpenVOS Commands Reference Manual* (R098). For information about other date/time commands, see the descriptions of the commands `set_jiffy_times`, and `check_jiffy_times` in this chapter.

`set_default_library_paths` ***Privileged***

Purpose

The `set_default_library_paths` command changes the list of directory path names that define the specified library. The changes are effective for all new processes on a module.

You use this command mainly in the `module_start_up.cm` file.

Display Form

```
------------------------- set_default_library_paths -------------------------
library_name:        include
library_path_names:
-module:             current_module
-check:              yes
-ask:                yes
```

Command-Line Form

```
set_default_library_paths [library_name]
            [library_path_names] ...
            [-module module_name]
            [-no_check]
            [-no_ask]
```

Arguments

▶ *`library_name`* CYCLE

The name of the library whose list of default library paths is to be changed. Possible values are `include`, `object`, `command`, and `message`. The default value is `include`.

▶ *`library_path_names`*

One or more path names of directories. You can specify the path name of any of these libraries by using the command function `'(current_dir)'` or `'(home_dir)'`; if the value for the *`library_name`* argument is `message`, you can also include the command function `'(referencing_dir)'` or `'(language_name)'`. The command functions must be enclosed in apostrophes.

OpenVOS uses the path names and command functions named in this argument as the list of library paths for the specified library for all new processes on the specified module. The names are placed on the list in the order specified. To display the list, issue the `list_default_library_paths` command.

If you omit a value for this argument, the current path names for the specified library are deleted and replaced with `(current_dir)` and `(master_disk)>system>`*`library_name_library`*, respectively.

If you specify the null string as a value for this argument, OpenVOS deletes the current path names for the specified library.

▶ `-module` *`module_name`*

Specifies the module whose list of library paths is to be changed. If you omit this argument, OpenVOS assumes the current module.

▶ `-no_check` CYCLE

Does not check that each value specified for the *`library_path_names`* argument is a valid object name. The default value is `yes`.

When you include this command in the `module_start_up.cm` file, specify the `-no_check` argument. Otherwise, if an object named as a library path name is on another system, the startup fails because the object is not known to the module.

▶ `-no_ask` CYCLE

Suppresses the prompt that asks you whether to delete all current default library path names when you specify an empty or null string for *`library_path_names`*. The default is to ask you before deleting library paths that contain objects. If you specify an empty or null string, the command suppresses the prompt and removes the library paths.

Explanation

A *library* is a set of one or more directories in which OpenVOS looks for objects of a particular type. There are four libraries.

- the `include` library, in which the compilers and the assembler look to find include files—For example, the OpenVOS PL/I compiler looks in the `include` library for files named in `%include` statements in OpenVOS PL/I programs.
- the `object` library, in which the `bind` command looks to find subroutines that are to be bound with the object module named in the command
- the `command` library, in which the OpenVOS command processor looks to find external commands—(See the `help` command display for a list of the external commands.)
- the `message` library, in which the command processor searches for per-command message files and help files

One of each of these libraries is available in the `(master_disk)>system` directory on each module for all processes running on the module. In addition, users can define their own libraries.

When OpenVOS looks for an object in a library, it refers to a list of directory path names that are known as the library paths for that library. It first searches for the object in the initial directory on the list. If it does not find the object there, it searches each subsequent directory in turn.

The library paths are contained in lists that you can maintain by issuing the following commands.

```
set_default_library_paths
add_default_library_path
delete_default_library_path
```

Using these commands, you can change this list for all new processes in a module. To display the list, issue the `list_default_library_paths` command. See the description of that command earlier in this chapter for an example of default library path lists.

If you issue the `set_default_library_paths` command with no value for the *`library_path_name`* argument, OpenVOS deletes the current default library paths and replaces them with two path names: `(current_dir)` and `(master_disk)>system>`*`library_name_library`*. For example, issuing the command `set_default_library_paths include -module m6` sets the following default include library paths.

```
(current_dir)
%s#m6>system>include_library
```

If you issue the `set_default_library_paths` command and specify the null string as a value for the *`library_path_name`* argument, OpenVOS deletes the current path names for the specified library without replacing them. For example, the command `set_default_library_paths include '' -module m6` deletes the current default include library paths.

By specifying the `-ask` argument with `set_default_library_paths`, you can prevent this from happening unintentionally. In either of the cases described in the preceding paragraphs, before deleting the existing default library paths, OpenVOS displays a prompt asking you to verify your intention. The prompt reads as follows:

```
Do you really want to clear the include library paths? (yes,
no)
```

If you specify the `-no_ask` argument, OpenVOS does not display the preceding prompt before deleting the path names.

You can override the default library paths for your own current process, and only for the duration of that process, by using the following commands (documented in the *OpenVOS Commands Reference Manual* (R098)).

```
add_library_path
delete_library_path
set_library_paths
```

For example, you can create a command macro and give it the same name as an OpenVOS command. If you want OpenVOS to execute the macro rather than the command when the name is specified, specify with a library path command that the directory containing the macro is at the top of the list of library paths in the command library.

When you use a command function in an argument value with a default library path command, enclose it in apostrophes (`'`). This prevents OpenVOS from evaluating the command function when you issue the library path command. Instead, it is evaluated and replaced by a path name (relative to the current process) when OpenVOS performs an actual search according to the most recently defined library paths.

Examples

The following command tells OpenVOS to search for a command library on the current module first in the current directory, then in the `(master_disk)>system>command_library` directory, and finally in the `(master_disk)>system>tools_library` directory.

```
set_default_library_paths command  '(current_dir)'
     >system>command_library >system>tools_library
```

Related Information

See the descriptions of the following commands in this chapter: `add_default_library_path`, `delete_default_library_path`, and `list_default_library_paths`. See the *OpenVOS Commands Reference Manual* (R098) for descriptions of the `add_library_path`, `delete_library_path`, `list_library_paths`, and `set_library_paths` commands. See also the *OpenVOS Commands User's Guide* (R089) for information about OpenVOS's search rules.

`set_default_time_zone` ***Privileged***

Purpose

The `set_default_time_zone` command changes the default time zones for one or more modules. The command updates the disk label and takes effect during the current bootload.

Display Form

```
----------------------------- set_default_time_zone -------------------------
time_zone:   █
difference:
-module:     current_module
-daylight:   no
```

Command-Line Form

```
set_default_time_zone time_zone
            [difference]
            [ -module module_name ...]
            [ -daylight]
```

Arguments

▶ *`time_zone`* **Required**

A code for the new time zone. Table 14-1 lists the time zone codes that OpenVOS recognizes. If you use one of these codes, you do not have to specify the *`difference`* argument.

You can enter the *`time_zone`* value in either uppercase or lowercase.

▶ *`difference`*

The number of minutes between the new time zone and Greenwich Mean Time. You can omit this argument if the new time zone is listed in Table 14-1.

▶ `-module` *`module_name`*

Specifies one or more names of modules whose time zone is to be changed. You can use star names. The default value is the current module.

▶ `-daylight` CYCLE

Sets the daylight time zone flag. By default, the daylight time zone flag is set to off.

Explanation

This command changes the current time zone of any process that has not set its own time zone. A user who is in a different time zone from that set for the module can put a `set_time_zone` command in his or her `start_up.cm` file.

Table 14-1 lists the time zones that are defined within OpenVOS, with the code and the *`difference`* value for each.

Table 14-1. Time Zones Supported in OpenVOS *(Page 1 of 3)*

Code	Description	Difference from GMT (in Minutes)	Implements Daylight Savings
`adt`	Atlantic Daylight	-180	yes
`ast`	Atlantic Standard	-240	no
`at`	Azores	-120	no
`bst`	British Summer	60	yes
`bt`	Baghdad	180	no
`cad`	Central Australia Daylight	630	yes
`cas`	Central Australia Standard	570	no
`cdt`	Central Daylight	-300	yes
`cet`	Central European	60	no
`chd`	China Daylight	540	yes
`cht`	China Time	480	no
`cst`	Central Standard	-360	no
`ead`	East Australia Daylight	660	yes
`eas`	East Australia Standard	600	no
`edt`	Eastern Daylight	-240	yes
`eet`	Eastern Europe	120	no
`est`	Eastern Standard	-300	no
`fst`	French Summer	120	yes

Table 14-1. Time Zones Supported in OpenVOS *(Page 2 of 3)*

Code	Description	Difference from GMT (in Minutes)	Implements Daylight Savings
`fwt`	French Winter	60	no
`gmt`	Greenwich Mean	0	no
`gst`	Greenland Standard	-180	no
`hdt`	Hawaii-Aleutian Daylight	-540	yes
`hfe`	Heure Francaise d’Ete	120	yes
`hfh`	Heure Francaise d’Hiver	60	no
`hkt`	Hong Kong	480	no
`hst`	Hawaii-Aleutian Standard	-600	no
`ist`	Indian Standard	330	no
`jst`	Japan Standard	540	no
`jt`	Java	450	no
`kdt`	Alaska Daylight	-540	yes
`kst`	Alaska Standard	-600	no
`mas`	Malaysia Standard	480	no
`mdt`	Mountain Daylight	-360	yes
`mes`	Middle Europe Summer	120	yes
`met`	Middle Europe	60	no
`mew`	Middle Europe Winter	60	no
`mst`	Mountain Standard	-420	no
`ndt`	Newfoundland Daylight	-150	yes
`nst`	Newfoundland Standard	-210	no
`nt`	Nome	-660	no
`nzd`	New Zealand Daylight	780	yes
`nzs`	New Zealand Standard	720	no
`pdt`	Pacific Daylight	-420	yes

Table 14-1. Time Zones Supported in OpenVOS *(Page 3 of 3)*

Code	Description	Difference from GMT (in Minutes)	Implements Daylight Savings
`phs`	Philippine Standard	480	no
`pst`	Pacific Standard	-480	no
`sdt`	Samoa Daylight	-660	yes
`sis`	Singapore Standard	480	no
`sst`	Samoa Standard	-600	no
`tpe`	Taiwan Standard	480	no
`tst`	Thailand Standard	420	no
`ut`	Universal	0	no
`wad`	West Australia Daylight	540	yes
`was`	West Australia Standard	480	no
`wat`	West Africa	-60	no
`wib`	Indonesia Standard	420	no
`ydt`	Yukon Daylight	-480	yes
`yst`	Yukon Standard	no	
`z`	Zulu (Universal)	0	no

Examples

The following command sets the default time zone to Central Standard Time on all modules in the current system.

```
set_default_time_zone cst -module m*
```

Related Information

See the description of the `set_time_zone` command in the *OpenVOS Commands Reference Manual* (R098).

`set_jiffy_times` ***Privileged***

Purpose

The `set_jiffy_times` command synchronizes with the current module the date and time on all modules in a system and then displays for each module the difference between the previous setting and the new setting. You can specify the interval at which synchronization occurs.

NOTE

Use of Network Time Protocol (NTP), which OpenVOS supports, synchronizes time with greater accuracy on all modules in a system. Do not use `set_jiffy_times` on modules that use NTP, as it will degrade NTP time synchronization. See “Time Synchronization Guidelines” on page 12-3 for related information.

Display Form

```
-------------------------------- set_jiffy_times -----------------------------
-verify_change:        no
-interval:
-from_remote_module:
```

Command-Line Form

```
set_jiffy_times [-verify_change]
            [-interval time]
            [-from_remote_module module_name]
```

Arguments

▶ `-verify_change` CYCLE

Verifies the changed interval. The default is `no`, meaning the interval is not verified.

- `-interval` *`time`*

  Specifies the length of time, in seconds, between clock synchronizations. The minimum amount of time is 5 seconds; the maximum is 32,767 seconds.

- `-from_remote_module` *`module_name`*

  Specifies the name of a time-server module on another system. A *time-server module* is a module that you designate to keep the official time for one or more systems connected by StrataNET or Open StrataNET.

Explanation

The information displayed by this command shows you how closely synchronized the times were **before** you issued the command. For information about the newly set times, issue the `check_jiffy_times` command. If you specify an interval, the `set_jiffy_times` command synchronizes the clocks at that interval. If you do not specify an interval, the clocks are synchronized once.

To synchronize the time in a multisystem configuration, specify the `-from_remote_module` argument. This argument enables you to designate one module to serve as a time-server module that keeps the official time for one or more systems connected by StrataNET or Open StrataNET. When you specify this argument, the command extracts the time from the specified time-server module and broadcasts it across the current system.

NOTE

The date and time on a module with a nonfunctioning or incorrectly functioning clock typically fall within the two-week time period starting at 00:00:00 on January 1, 1980 (the default time and date of a module without a clock). If `set_jiffy_times` determines that the current module's date and time fall within this time period, it does not reset the clocks of the other modules on the system. Instead, it returns the following message.

```
Clocks on current system not set because current
module clock is incorrect.
```

Examples

The output of the following command shows the time to which the modules were set by this command, and the time on each module when the times were synchronized.

```
ready: set_jiffy_times
MODULE      TIME SET TO                TIME WAS                DELTA
m3    21-11-26 13:33:08.4769  21-11-26 13:33:08.5791 +0.1021 sec
m6    21-11-26 13:33:08.4769  21-11-26 13:33:09.3273 +0.8503 sec
m10   21-11-26 13:33:08.4769  21-11-26 13:33:09.2626 +0.7856 sec
m14   21-11-26 13:33:08.4769  21-11-26 13:33:08.6470 +0.1700 sec
m21   21-11-26 13:33:08.4769  21-11-26 13:33:09.1755 +0.6985 sec
m27   21-11-26 13:33:08.4769  21-11-26 13:33:08.8416 +0.3646 sec
m49   21-11-26 13:33:08.4769  21-11-26 13:33:08.9187 +0.4418 sec
m72   21-11-26 13:33:08.4769  21-11-26 13:33:11.7119 +3.2349 sec
m75   21-11-26 13:33:08.4769  21-11-26 13:33:08.9209 +0.4439 sec
m80   21-11-26 13:33:08.4769  21-11-26 13:33:08.9898 +0.5129 sec
m101  21-11-26 13:33:08.4769  21-11-26 13:33:08.9363 +0.4593 sec
m111  21-11-26 13:33:08.4769  21-11-26 13:33:08.3616 -0.1153 sec
m113  21-11-26 13:33:08.4769  21-11-26 13:33:07.6708 -0.8061 sec
m114  21-11-26 13:33:08.4769  21-11-26 13:33:08.9201 +0.4431 sec
m115  21-11-26 13:33:08.4769  21-11-26 13:33:08.7683 +0.2913 sec
m17   21-11-26 13:33:08.4769  21-11-26 13:33:09.6608 +1.1838 sec
m24   21-11-26 13:33:08.4769  21-11-26 13:33:08.4752 -0.0017 sec
m13   21-11-26 13:33:08.4769  21-11-26 13:33:08.1699 -0.3070 sec
m25   21-11-26 13:33:08.4769  21-11-26 13:33:08.7620 +0.2850 sec
m26   21-11-26 13:33:08.4769  21-11-26 13:33:08.7639 +0.2869 sec
m28   21-11-26 13:33:08.4769  21-11-26 13:33:08.2929 -0.1840 sec
```

A value in the `DELTA` column represents the difference between the time on the current module and the time on the target module **before** either of them were set. The values are in seconds so they are consistent with the `check_jiffy_times` command.

An asterisk (`*`) to the right of the module name means that `set_jiffy_times` could not set the time on that particular module and the remainder of the information about that module should be ignored. Try issuing `set_jiffy_times` a second time to correct the situation.

To propagate the time in a multisystem configuration connected by StrataNET or Open StrataNET, specify the `-from_remote_module` argument. For example, the following command propagates the time from the `%sales#m1` time-server module to the `%dev` system. You must be logged in to a `%dev` module when you issue this command.

```
set_jiffy_times -from_remote_module %sales#m1
```

Related Information

See the descriptions of the commands `check_jiffy_times` and `set_date_time` in this chapter.

See ‘‘Time Synchronization Guidelines’’ on page 12-3 for the preferred way to synchronize an OpenVOS module with an Internet time service.

`set_lock_wait_time` ***Privileged***

Purpose

The `set_lock_wait_time` command sets, for each specified module, the default wait time for I/O operations that can return one of the following messages.

```
e$file_in_use      e$no_pipe_readers
e$key_in_use       e$no_pipe_writers
e$region_in_use    e$tp_wait_to_win
e$record_in_use
```

The wait time is the maximum number of seconds that a process or task waits to lock either a file or record during any I/O operation. If no program or process wait time has been set, the default lock wait time is used.

Display Form

```
------------------------------ set_lock_wait_time -------------------------
time:
-module: current_module
```

Command-Line Form

```
set_lock_wait_time time
           [-module module_name]
```

Arguments

- ▶ *`time`* **Required**

  The wait time to be set, in seconds. The value must be between 0 and 1000.

- ▶ `-module` *`module_name`*

  Specifies the name of the module whose wait time is to be set. You can specify a star name for this value. If you omit this argument, OpenVOS assumes the current module.

Explanation

The installation software includes in the `module_start_up.cm` file a `set_lock_wait_time` command that sets the wait time to 10 seconds. This should be an appropriate value for most modules. However, if the module has an unacceptable level of lock conflicts, you may want to change the wait time to a higher value.

If you have not set the wait time for a module, OpenVOS uses a default wait time of zero seconds.

Examples

The following command, as it might appear in a `module_start_up.cm` file, sets the wait time to 10 seconds.

```
set_lock_wait_time 10
```

Related Information

See the descriptions of `display_lock_wait_time` in this chapter and `display_process_lock_wait_time` and `set_process_lock_wait_time` in the Transaction Processing Facility manuals. See the *OpenVOS Commands Reference Manual* (R098) for an explanation of locking.

set_scheduler_info

Purpose

The `set_scheduler_info` command sets for the current bootload the parameters that OpenVOS will use to schedule processing time for either interactive or batch processes on a specified module.

Display Form

```
----------------------------- set_scheduler_info ----------------------------
priority:
-type:
-count:
-time:
-process: interactive
-socket:  1
-module:  current_module
```

Command-Line Form

```
set_scheduler_info priority
         -type quantum_type
         -count quantum_count
         -time quantum_time
         [-process process_type]
         [-socket number]
         [-module module_name]
```

Arguments

▶ *priority* **Required**

The priority level. OpenVOS's priority levels are 0 through 9.

▶ `-type` *quantum_type* **Required**

Specifies a list of one to four numbers. Each number represents a time-slice type that the process can receive. Number values must range from `1` to `255`. The number value of `type` determines the relative priority, with a type-1 time slice having the highest priority relative to the other time-slice types, and a type-255 time slice having the lowest priority.

The number of entries (one to four entries) in this list determines the number of entries that must be specified in the lists of the `-count` and `-time` arguments.

▶ `-count` *`quantum_count`* **Required**

Specifies a list of one to four numbers. Each number corresponds to the time-slice type in the same position in the `-type` list and represents the number of time slices of the corresponding type that a process can receive. Number values must range from `1` to `16`.

This list must contain the same number of entries (one to four entries) as specified in the lists of the `-type` and `-time` arguments.

▶ `-time` *`quantum_time`* **Required**

Specifies a list of one to four numbers. Each number corresponds to the time-slice type in the same position in the `-type` list, and represents the number of milliseconds in the corresponding time-slice type. Valid values for the numbers in this list are from `2` to `4000`.

This list must contain the same number of entries (one to four entries) as specified in the lists of the `-type` and `-count` arguments.

▶ `-process` *`process_type`* CYCLE

Specifies the type of process for which the command will set the scheduling parameters. The values are `interactive` and `batch`. The default is `interactive`.

▶ `-socket` *`number`*

Specifies the number of the StrataLINK sockets to be associated with the specified priority level and process type. The default is `1`.

NOTE

If you issue the `set_scheduler_info` command with the `-socket` argument on modules connected by Open StrataLINK, this argument has no effect.

▶ `-module` *`module_name`*

Specifies one or more modules for which the command will set the scheduling parameters. This value can be a star name. The default is the current module.

Explanation

Since `set_scheduler_info` is effective only for the current bootload, you must include it in the `module_start_up.cm` file if you want changes to the default scheduling parameters to remain in effect.

Changing the full set of scheduling parameters requires 20 `set_scheduler_info` commands, since each process type (batch and interactive) has 10 priority levels (0-9).

Examples

The following command sets the parameters for interactive priority level `4` on all modules whose names begin with `m`.

```
set_scheduler_info 4 -type 2 4 6 10 -count 1 3 8 1
     -time 1000 2000 2000 1000 -module m*
```

Related Information

See Chapter 2, ‘‘Setting and Defining Priority Levels,’’ and the description of the `display_scheduler_info` command earlier in this chapter.

set_system_log_mode

Purpose

The `set_system_log_mode` command sets the system log mode for the current process so that error messages are entered into the system error log. When the `state` argument is set to on, error messages are put in the `(master_disk)>system>syserr_log.(date)` file.

You use this command in the `module_start_up.cm` file to log any messages generated during the startup.

Display Form

```
------------------------------ set_system_log_mode --------------------------
state: on
```

Command-Line Form

```
set_system_log_mode [state]
```

Arguments

▶ `state` CYCLE

Enables or disables logging. In the command-line form, *`state`* can be `on` or `off`. The default is `on`.

Examples

The following command sets the system log mode to `off` for the current process.

```
set_system_log_mode off
```

Related Information

See the manual *OpenVOS System Administration: Starting Up and Shutting Down a Module or System* (R282) for related information.

set_tuning_parameters *Privileged*

Purpose

The `set_tuning_parameters` command allows you to set the system tuning parameters of a specified module for the duration of a bootload.

Display Form

```
----------------------------- set_tuning_parameters -------------------------
-module:                        current_module
-max_events_per_process:        current_value
-bm_max_buffers:                current_value
-bm_min_buffers:                current_value
-bm_max_virtual_pages:          current_value
-bm_modified_grace_time:        current_value
-cm_transient_mod_grace_time:   current_value
-bm_unreferenced_grace_time:    current_value
-bm_referenced_grace_time:      current_value
-bm_free_grace_time:            current_value
-cm_cache_mem_pct:              current_value
-cm_max_resident_pct:           current_value
-cm_min_cache_priority:         current_value
-cm_disk_mod_limit:             current_value
-cm_disk_write_limit:           current_value
-cm_cate_write_limit:           current_value
-recover_disk_priority:         current_value
-unused_dir_timeout:            current_value
-max_events_per_task:           current_value
-max_events_per_module:         current_value
-max_local_devices_per_module:  current_value
```

Command-Line Form

```
set_tuning_parameters [-module module_name]
            [-max_events_per_process max_limit]
            [-bm_max_buffers max_number_buffers]
            [-bm_min_buffers min_number_buffers]
            [-bm_max_virtual_pages max_number_pages]
            [-bm_modified_grace_time seconds]
            [-cm_transient_mod_grace_time seconds]
            [-bm_unreferenced_grace_time seconds]
            [-bm_referenced_grace_time seconds]
            [-bm_free_grace_time seconds]
            [-cm_cache_mem_pct percentage]
            [-cm_max_resident_pct percentage]
            [-cm_min_cache_priority priority]
            [-cm_disk_mod_limit max_number_mod_blocks]
            [-cm_disk_write_limit max_number_write_requests]
            [-cm_cate_write_limit max_number_write_requests]
            [-recover_disk_priority number]
            [-unused_dir_timeout seconds]
            [-max_events_per_task max_number_events]
            [-max_events_per_module max_number_events]
            [-max_local_devices_per_module max_number_devices]
```

Arguments

▶ `-module` *`module_name`*

Specifies the module for which the tuning parameters apply. You cannot change this value in the display form. If you omit this argument in the command-line form, the default is the current module.

▶ `-max_events_per_process` *`max_limit`*

Specifies the maximum number of user events allowed per process on the module. The default value depends on the memory configuration and is set internally at bootload. If you specify a value that is inconsistent with the memory configuration, even if it is within the boundaries of allowed values, the command uses the internally set value instead. By default, this value is set to 8,704 at bootload. The maximum value allowed is 32,767; the minimum value is 16.

▶ `-bm_max_buffers` *`max_number_buffers`*

Specifies the maximum number of physical disk cache buffers (physical pages in the cache). The default value depends on the memory configuration and is set internally at bootload. The maximum value allowed depends on the module's memory configuration and cannot use more memory than that calculated from the formula described in the description of the `-cm_cache_mem_pct` argument; the minimum value is 32. If you specify a value that is inconsistent with the memory configuration, even if it is within the boundaries of allowed values, the command uses the internally set value instead. This value must be greater than or equal to the values specified in `-bm_min_buffers` and `-bm_max_virtual_pages`.

You may need to change this value if you have a large configuration or if you want to dedicate large amounts of physical memory to the disk cache.

- ► `-bm_min_buffers` *`min_number_buffers`*

  Specifies the minimum number of physical disk cache buffers. By default, this value is set to half of the value of the `-bm_max_buffers` argument at bootload. This value must be greater than 32 and less than or equal to the value specified in `-bm_max_buffers`.

  When the number of unused buffers exceeds this value, the unused buffers are returned to the OpenVOS memory pool and are available for general reuse after expiration of the period of time specified by the *`bm_free_grace_time`* argument. The buffers are reallocated when needed again for cache usage.

- ► `-bm_max_virtual_pages` *`max_number_pages`*

  Specifies the maximum number of virtual pages used to map the physical pages in the cache. Typically, you do not need to change this value. When additional virtual memory is needed for other purposes, you can reduce *`max_number_pages`*. When virtual memory usage allows, increasing *`max_number_pages`* may improve cache performance. This value is limited to no more than 50% of the available virtual memory.

  If you specify a value that is inconsistent with the memory configuration, even if it is within the boundaries of allowed values, the command uses the internally set value instead. This value must be less than or equal to the value specified in `-bm_max_buffers`.

- ► `-bm_modified_grace_time` *`seconds`*

  Specifies the length of time that the cache manager waits after a block is modified before the cache manager writes the block to disk. (The cache manager waits the number of specified *`seconds`* after the last reference, but never more than three times the number of specified *`seconds`* from the first reference.) The default value, 60 seconds, is adequate for most purposes. You can reduce the value if the system's workload is such that the system will not rewrite a record in that amount of time (for example, if the system is updating random data records in a very large database). You can increase the value to reduce the write I/O rate if the workload makes multiple updates to records within a given time. OpenVOS provides no tools to determine this workload; you must determine it yourself or analyze the application.

- `-cm_transient_mod_grace_time` *`seconds`*

  Specifies the length of time that the cache manager waits after a block from a transient file is modified before the cache manager writes the block to disk. For transient files, a modified block is not written until it has expired, even if force writing is requested by the cache manager interface (as may be the case with the target file of the `copy_file` command.)

  The value *`seconds`* is limited to the values 0 through 3600. By default, this value is set to 2 at bootload. The value of *`seconds`* is limited by the value of the `-bm_modified_grace_time` argument. You can specify a higher value, but transient blocks will expire when the period of time specified by the `-bm_modified_grace_time` argument has elapsed.

- `-bm_unreferenced_grace_time` *`seconds`*

  Specifies the length of time, in seconds, before an unreferenced physical page is available for reuse as a buffer for a different disk block. By default, this value is set to 300 seconds at bootload.

- `-bm_referenced_grace_time` *`seconds`*

  Specifies the length of time, in seconds, that the cache manager waits before forcing a referenced block to become unreferenced. By default, this value is set to 60 seconds at bootload.

- `-bm_free_grace_time` *`seconds`*

  Specifies the length of time, in seconds, before an unreferenced physical page is returned to the pool of physical pages available for virtual memory management. By default, this value is set to 300 seconds at bootload.

- `-cm_cache_mem_pct` *`percentage`*

  Specifies the percentage of memory over 128 MB to use for the cache. This value may need to be increased from its default of 60 if the `dump_cache_info` request shows that the real max cache size is lower than the value specified in `-bm_max_buffers`.

  The value *`percentage`* is the percentage of memory beyond the first 128 MB that is used for the cache. The cache manager can use an amount of physical memory that is determined by the following formula:

      20% of the first 64 MB of physical memory in the machine
    + 40% of the next 64 MB, plus
    + *N*% of the remaining physical memory

  By default, this value is set to 60 at bootload. The value of *`percentage`* is limited to the values 0 through 95.

- `-cm_max_resident_pct` *`percentage`*

  Specifies the percentage of the maximum cache size to make available for memory-resident files. By default, this value is set to 50 at bootload. The value of *`N`* is limited to the values 0 through 95.

- `-cm_min_cache_priority` *`priority`*

  Specifies a minimum priority for processes that are entitled to retain data in the cache. The value *`N`* is the priority that is compared to the process's priority and must be a value of 0 (the default) through 9. Blocks used by processes running below priority *`N`* are the first to expire if the cache usage is above the minimum.

- `-cm_disk_mod_limit` *`max_number_mod_blocks`*

  Specifies the maximum number of modified blocks per disk that the cache manager is allowed to hold in cache before initiating disk writes. When this limit is reached, blocks are queued for writing even if their mod expiration time has not been reached. Processes making the modifications may be slightly delayed in order to accomplish this. This value does not provide a hard limit, and the actual number of modifications may exceed it by 5% to 10%. The value of *`max_number_mod_blocks`* is limited to the values 1 through 32,767. By default, this value is set to 4096 at bootload.

- `-cm_disk_write_limit` *`max_number_write_requests`*

  Specifies the maximum number of write requests that the cache manager is allowed to queue to each disk. When this limit is reached, future queue writes are delayed until the disk queues are reduced to 50% of this value. The value of *`max_number_write_requests`* is limited to the values between 1 and 32,767. By default, this value is set to 1024 at bootload.

- `-cm_cate_write_limit` *`max_number_writes`*

  Specifies the maximum number of writes that are allowed to be concurrently queued for blocks associated with a cate before delays will occur for processes attempting to subsequently access other blocks in that cate. (A *cate* is an OpenVOS data structure.)

  The value of *`max_number_writes`* is limited to the values 1 through 32,767. By default, this value is set to 32,767 at bootload, and this value is treated as unlimited. Do not change this to a lower value; the ability to do so is provided only for compatibility purposes.

  NOTE

  This value is no longer used; it appears only for compatibility purposes.

- `-recover_disk_priority` *`number`*
  Specifies the priority at which `recover_disk` will run. By default, this value is set to 1 at bootload. The allowed values are 0 to 9.

- `-unused_dir_timeout` *`seconds`*
  Specifies the length of time a directory can be unused before its contents are written to disk. By default, this value is set to 120 seconds at bootload.

- `-max_events_per_task` *`max_number_events`*
  Specifies the maximum number of events that a task on the module can wait for. The default value is 8,704. The maximum value is 32,767. Note that a single-task process can wait for up to the limit of events set by the `-max_events_per_process` argument.

- `-max_events_per_module` *`max_number_events`*
  Specifies the maximum number of events allowed on the module. The value `0` (the default) has no effect on the limits (that is, OpenVOS ignores the argument). The maximum value is 524,287.

- `-max_local_devices_per_module` *`max_number_devices`*
  Specifies the maximum number of local devices allowed on the module. The value you specify must be equal to or greater than the current size of the device table plus 10. The default value is 70,000. The maximum value is 200,000.

Explanation

The `set_tuning_parameters` command allows you to set certain parameters on a specified module. The following table shows the default values at bootload for these parameters. Any modifications made are valid until the next time you either boot the module or you modify the parameters again using `set_tuning_parameters`. When you use the display form, the information displayed for each parameter is the module's current value for that parameter.

NOTE

The cache manager does not accept values for `set_tuning_parameters` command arguments that are above or below the maximum and minimum values that the cache manager allows. However, you do not receive an error message if you specify a value above or below the maximum and minimum values that the cache manager allows.

You can place the `set_tuning_parameters` command in the `module_start_up.cm` file to change the parameters for a specific bootload. If you change a parameter's value, that value becomes the default for the duration of the current bootload only.

Argument	Default Value
`-max_events_per_process`	`8704`
`-bm_max_buffers`	System-dependent
`-bm_min_buffers`	Half of the value of `-bm_max_buffers`
`-bm_max_virtual_pages`	System-dependent
`-bm_modified_grace_time`	`60`
`-cm_transient_mod_grace_time`	`2`
`-bm_unreferenced_grace_time`	`300`
`-bm_referenced_grace_time`	`60`
`-bm_free_grace_time`	`300`
`-cm_cache_mem_pct`	`60`
`-cm_max_resident_pct`	`50`
`-cm_min_cache_priority`	`0`
`-cm_disk_mod_limit`	`4096`
`-cm_disk_write_limit`	`1024`
`-cm_cate_write_limit`	`32,767` (unlimited)
`-recover_disk_priority`	`1`
`-unused_dir_timeout`	`120`
`-max_events_per_task`	`8704`
`-max_events_per_module`	`524287`
`-max_local_devices_per_module`	`70000`

Related Information

You can view the tuning parameters by issuing the `display_tuning_parameters` command, documented earlier in this chapter. For information about optimizing the tuning parameters on your module, see Chapter 13, ‘‘Tuning the Disk Cache.’’

site_call_system

Purpose

The `site_call_system` command invokes the Site Call System subsystem that you use to access the site call database at your site.

NOTE

If you are using the Remote Service Network (RSN) over the X.25 communications protocol, see the *VOS Communications Software: X.25 and StrataNET Administration* (R091) manual.

Display Form

```
---------------------------- site_call_system ------------------------------
-request_line: █
-quit:         no
-omit_deleted: no
-show_audit:   no
```

Command-Line Form

```
site_call_system [-request_line request_line]
            [-quit]
            [-omit_deleted]
            [-show_audit]
```

Arguments

▶ `-request_line` *`request_line`*

Specifies one or more Site Call System request strings, separated by semicolons (`;`). A request string consists of the name of the request and any arguments you specify. In the command-line form, a value for *`request_line`* that contains spaces must be enclosed by apostrophes (`'`). The following requests are available.

`add_call`	*`list_calls`*	*`remove_court_value`*
`add_court_value`	*`list_court_values`*	*`rsn_mail`*
`change_call`	*`maint_request`*	*`set_mode`*
`display_call`	*`new_call`*	*`update_calls`*
`help`	*`purge_calls`*	
	`quit`	

Once the requests you specify in the *`request_line`* argument finish executing, your process either goes to Site Call System request level, signified by the prompt `scs>`, or goes back to command level, depending on the value of the `-quit` argument. If you do not specify a value for *`request_line`*, this occurs immediately.

▶ `-quit` CYCLE

Returns to command level once the requests specified in the *`request_line`* argument (if any) finish executing. The default is `no`, which sends your process to Site Call System request level once the requests have executed.

▶ `-omit_deleted` CYCLE

Does not display deleted calls when you invoke the `list_calls` request. The default is `no`, which means deleted calls are not omitted when you invoke `list_calls`.

See the `list_calls` `-status` and `-site_status` argument descriptions, later in this command description, for more information.

▶ `-show_audit` CYCLE

Displays audit text when you invoke the `update_calls` request. The default is `no`, which means audit text is not displayed when you invoke `update_calls`.

See the `update_calls` request description, later in this command description, for more information.

Explanation

The Site Call System subsystem provides an easy-to-use environment for accessing a call database at your site. You can use it to track calls and communicate with the HUB; you can also use it to access certain Remote Service Network (RSN) functions, such as the `maint_request` command.

Within this subsystem, your calls are stored in the directory `(master_disk)>system>site_call_system` in two database files named `site_call_db` and `site_call_text`. This directory is known as the *site call database directory*. The site call database directory and the database files it contains are not provided as part of the installation process. Instead, they are created automatically the first time you execute `site_call_system` on your module.

On a multimodule system, create one `site_call_system` directory and then create links between that directory and the other modules.

In addition to database files, the site call database directory may contain some ancillary files related to the operation of the Site Call System subsystem. These files do not require any management on your part.

For the Site Call System to operate properly, a process called SiteCallXfer must be running. This process executes the `site_call_xfer.pm` program that receives call updates from the HUB and posts them to the local site call database. Without it, you can escalate (send) local calls to the HUB, but the HUB cannot escalate calls to your site.

The SiteCallXfer process is **not** started automatically. After the site call database directory and its contents are created when you first execute the `site_call_system` command, you must start the SiteCallXfer process by invoking `start_site_call_xfer` program. See the description of the `start_site_call_xfer` command later in this chapter.

The SiteCallXfer process is then started when the `start_rsn` command is invoked from the `mddmon_start_up.cm` file (which, in turn, is started by the `start_stcp.cm` file) on the bridge module at reboot. The `start_rsn` command does not start SiteCallXfer directly; it invokes the `start_site_call_xfer` command to start the process.

NOTE

The `start_site_call_xfer` program starts the SiteCallXfer process **only** if the `site_call_db` file (which is located in the directory `(master_disk)>system>site_call_system`) already exists, and if the command specifies the bridge module. Therefore, invoking `start_site_call_xfer` before the database directory and contents are created, either directly or via `start_rsn`, does not start the SiteCallXfer process.

To use the Site Call System subsystem, the following must have modify access and write default access to the site call database directory.

- all users of `site_call_system`
- the `rsn_server` process, which runs `rsn_server.pm` to report hardware calls
- the SiteCallXfer process, which runs `site_call_xfer.pm` to manage incoming updates from the HUB

To enter the Site Call System subsystem, issue the `site_call_system` command. The prompt `scs>` signifies that your process is at request level. At request level, you invoke Site Call System requests the same way as for OpenVOS commands, using either the command line or the display form. Each request is explained later in this command description.

You can also execute OpenVOS internal commands without exiting the Site Call System subsystem. At the `scs>` prompt, type two periods (`..`), followed by the name of the command(s) you want to execute. For example, the following request changes the current directory to the user's home directory, lists all of its subdirectories, then returns the process to subsystem request level.

```
..change_current_dir;list -dirs
```

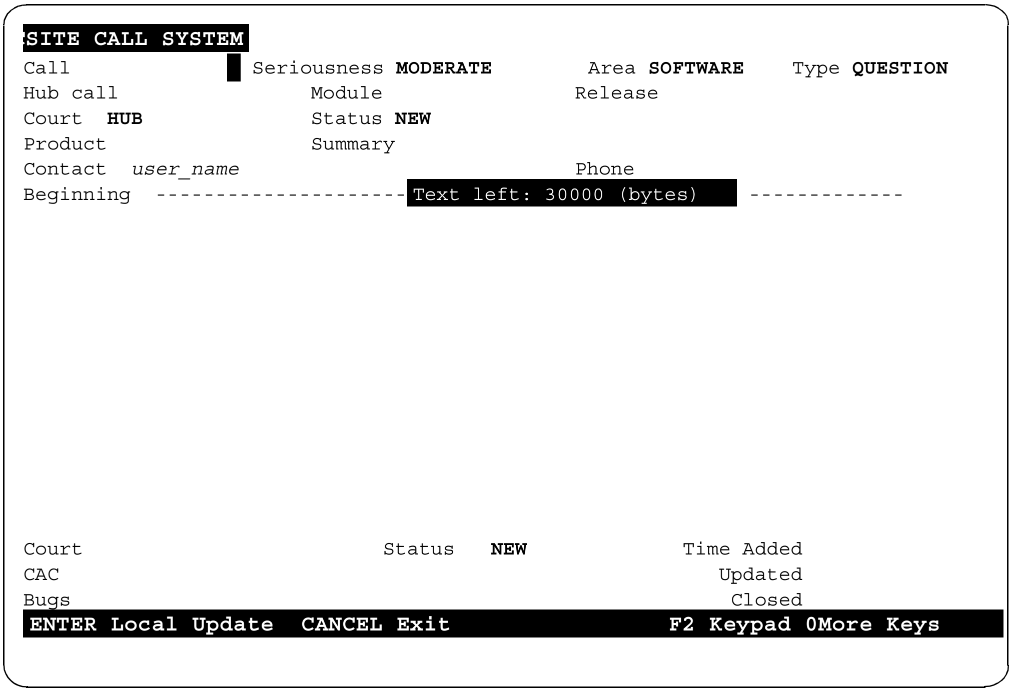
Note the following:

- Local calls originate from your site; HUB calls originate from the HUB.
- Local and HUB calls are identified by call numbers assigned automatically by OpenVOS.
  - A *local call number format* has an integer value. The first call entered into your site's call database is number 1; subsequent call numbers are assigned sequentially.
  - HUB call numbering uses a composite form called a hybrid call number format that consists of an integer value prefixed, optionally, by an alphabetic character. The prefix identifies the home HUB of the call: the HUB in the Redundant Call Database System (RCBD) network that entered the call into the global HUB database. If the HUB that services your site is not part of an RCDB network, the calls it originates may not use a prefix. The HUB can tell you the correct prefix (if any) to use when referencing a HUB call.
- The term *default call number* refers to the local call number most recently used in the current Site Call System session.

- You can use the OpenVOS abbreviations facility to create abbreviations for Site Call System request names and arguments.

The Site Call Form

The site call form is a standard Forms Management System form. (See the Forms Management System manuals for more information on forms.) Use it to either change or add local calls to the database. It appears on your terminal when you issue either the `change_call` request or the `new_call` request. The initial form of the Site Call System subsystem follows.



```
SITE CALL SYSTEM
Call                 █ Seriousness MODERATE        Area SOFTWARE    Type QUESTION
Hub call                    Module                Release
Court  HUB                  Status NEW
Product                     Summary
Contact  user_name                                Phone
Beginning  --------------------Text left: 30000 (bytes)   -------------












Court                          Status   NEW           Time Added
CAC                                                      Updated
Bugs                                                      Closed
 ENTER Local Update  CANCEL Exit                 F2 Keypad 0More Keys
```

Each field in the form corresponds to a field in a call's database entry.

Site Information

Site information is found on the first five lines. This section contains information set locally at your site and is known as the call header. Press the ←Enter key to move the cursor from field to field; press the ← and → keys to cycle the values.

▶ `Call`

The local call number. When you issue the `new_call` request and perform a local update on the call, a number is automatically assigned. When you use the `change_call` request, the number in this field is either the number you specified in the `call_no` argument (which may be the default call number), or the local call number corresponding to the HUB call number you specified in the `-hub_call` argument. You can change this number and use the `Read Call` function if you want to display and/or update information about another call in the database.

- `Seriousness`
  The degree of seriousness of the call. See Table 14-4 for values and descriptions.

- `Area`
  The subject of the call. See Table 14-3 for values and descriptions.

- `Type`
  The purpose of the call. See Table 14-5 for values and descriptions.

- `Hub Call`
  The HUB call number corresponding to the local call number displayed. You can change this number and use the `Read Call` function if you want to display information about another call in the database.

  When inserting a value into this field, you must use the call number format specified by the HUB servicing your system. See the description of the hybrid call format, earlier in this command description, for details.

- `Module`
  The name of the module about which the call is made. By default, the system uses the current module.

- `Release`
  The release of OpenVOS running on the module specified in the `Module` field. If you do not provide a value for this field, the release of OpenVOS running on the current module is used when you enter the call information into the database. However, if you change the `Module` field and press the ←Enter key without first clearing the `Release` field, the value is not updated accordingly.

- `Court`
  The person or group at your site responsible for the next action on the call. Issue the `list_court_values` request, documented later in this command description, to see a list of allowed site court values.

  See "Configuring Site Court Values" on page 14-134 for information on how to create custom site court values.

- `Status`
  The site status of the call. Table 14-2 for values and descriptions. (HUB status is different from site status; see "HUB Information" on page 14-131.)

- ▶ `Product`

  The name of the product about which the call is made. A value is not required for this field, and there is no default.

- ▶ `Summary` **Required**

  A brief description of the purpose of the call. You must supply a value in this field.

- ▶ `Contact`

  The name of the person at your site designated as the main contact regarding the current call number. If you do not specify a name, the name of the current user is used.

- ▶ `Phone` **Required**

  The phone number of the person specified in the `Contact` field. You must supply a number in this field.

Text

Text is the area under the heading **`Text left: 30000 (bytes)`**. In this section you enter the text of the update you want to make. Press the ←Enter key to move the cursor down in the buffer. When you reach the bottom of the text window, press the ←Enter key. The buffer scrolls down so that the bottom two lines of text become the top two lines in the window. You can continue filling in the buffer in this manner. After you finish typing, you have the following options, displayed on the status line.

- Exit the form, avoiding the update (the `Exit` function).
- Cancel the form, avoiding the update (the `CANCEL` function).
- Add a new call to the local database only (the `ENTER` function).
- Update the local database only with changes to an existing call (the `Local Update` function). This function allows you to keep your call transaction completely local.
- Update and escalate both this and previous updates to the HUB (the `Extended Escalation` function).
- Update and escalate this update only to the HUB (the `Escalation` function).
- Display more key functions for cursor movement and appending text.

When you update the call locally, the text is entered into a buffer containing the original call and any updates made to it. Each call is stored in a separate text buffer that can hold a maximum of 30,000 characters.

The beginning of the text section has two fields, modified by the Site Call System subsystem.

- `Beginning/More/End`

  Shows the relative position of the text window in the text buffer. This field has three values: `Beginning` indicates the first screen of text, `End` indicates the last screen of text, and `More` indicates the middle of the text buffer.

- `Text left: `*`N`*` (bytes)`

  Displays a number showing how many of the maximum 30,000 characters remain unused in the text buffer.

The file `site_call_text` stores the text you enter. It also stores another type of text, called *audit text*, which shows how the call has been changed, when, and by whom. A call acquires audit text whenever the value in one or more of its fields is changed.

Audit text of the following form accompanies any update of the call.

```
status_change by user_name (date) (time)
```

The *`status_change`* value describes the type of change to the site status of the call. If the field value is unchanged but the call is otherwise updated, the value shown for *`status_change`* is `Updated`; if the site status is changed, the value shown is the new value, for example `Opened` or `Closed`. The values for *`user_name`*, *`.date`*, and *`time`* are the name of the user who updated the call and the date and time of the update.

The audit text for a call that has been updated might read as follows:

```
Call    202: <updated>
Deleted by Lee_Smith 21-10-05 15:03:31 est.
Contact changed from '' to 'Lee_Smith'.
```

Changes to other fields produce an additional message that notes the name of the field and the old and new values. For example, a change to the HUB status of the call might be noted as follows:

```
Hub Status changed from OPEN to CLOSED.
```

HUB Information

HUB information is contained in the three lines beginning with `Court`. This section contains information received from the HUB and some related local time values.

- `Court`

  The location responsible for the next action on the call, for example `HUB` or `USER`.

- `Status`

  The HUB status of the call. This is not the same as the site status. (See "Site Information" on page 14-128.) This value can be a null string, or one of the following: `NEW`, `OPEN`, `CLOSED`, `DELETED`, or `UNRESOLVED`.

- `CAC`
  The person to contact at the HUB regarding the call.

- `Bugs`
  Any bugs associated with the call. Up to four may be listed.

- `Time Added`
  The date and time the call was added to the site's database.

- `Updated`
  The date and time the call was last updated at the site.

- `Closed`
  The date and time the call was closed at the site. If the call is still open, this field has no value. If the call is closed, the value is blinking.

When any of this information is updated at the HUB (for example, if the support engineer who is handling the call changes the status from `NEW` to `OPEN`), the site database is automatically updated.

Function Keys

The list of active function keys appears as a single line at the bottom of the screen. The key sequences shown in this section are for the V105 terminal with the PC/+ 106 keyboard and the V105/V109 with the EPC keyboard. The OpenVOS Software Release Bulletin provides the corresponding key sequences for the V105 terminal with an ANSI keyboard.

Not all of the actions appear on the initial screen; invoke the `More Keys` function to display more actions. The following functions are available.

- `Local Update`
  Updates the call in the local database only.

- `Exit`
  Returns to subsystem request level.

- `Extended Escalate`
  Updates the call in the local database. Then, along with the header and the text in the current update, it sends any text since the last escalation. If there was no previous escalation, it sends all text from the beginning of the text buffer (which includes any previous local updates).

- `More Keys`
  Displays function keys and actions in addition to those initially displayed on the line. The `More Keys` function key appears in the same location on the line with each different set of keys displayed. Pressing `More Keys` repeatedly rotates back to the original set of keys and actions displayed.

▶ `Read Call`

Reads an existing call from the database. The call selected is the one whose number is in the `call_no` field, unless the cursor is on the `hub_call` field. In this case, the call selected is the HUB call with that number. Note that if the specified HUB call is associated with a different local call, the number in the `call_no` field changes. The new local call number then becomes the default call number.

▶ `Append from File`

Prompts you for the path name of a file, and reads and appends the specified file to the end of the current call's text buffer.

NOTE

Once a call has been escalated to the HUB via the `site_call_system` command, the following marker is placed in the text to indicate this fact.

```
** Call sent up to the HUB.
```

If you intend to use that text in another call (for example, using the `-output_path` option with the `display_call` request), you must edit the text to alter the marker, or strip it out altogether. Otherwise, the `site_call_system` command treats the text as if it already has been escalated in the current call and will not re-escalate it. This caution applies when using the site call form or any of the following requests.

```
add_call
display_call
new_call
update_calls
```

▶ `Start New Call`

Clears the text buffer, and replaces with default values the current values in all fields. This action is the same as exiting the site call form and specifying the `new_call` request at request level.

▶ `Key Explanations`

Clears the form and replaces it with a help screen that displays the available keys and longer descriptions of the actions they perform. Press the $\boxed{*}$ key on the keypad to return to the site call form.

▶ `Scroll Up/Scroll Down`

Moves the text window up, toward the beginning of the text buffer, or down, toward the end of the text buffer.

▶ `Top of Text/Bottom of Text`

Moves the text window up, directly to the beginning of the text buffer, or down, directly to the end of the text buffer.

▶ `Partial Escalation`
  Updates the call in the local database. Then, the fields in the top section of the screen (the header), and only the text in the current update, are sent to the HUB via the RSN. Any text that was part of a previous local update is not sent to the HUB.

Configuring Site Court Values

The `Court` field in the Site Information section of the site call form, and the `-site_court` option of various other `site_call_system` requests, allow you to specify who, at your site, is responsible for the next action on a given call.

The `site_call_system` command provides a few default values for this field: `HUB`, `SITE`, `ENGINEER`, and `USER`. However, because every site has individual needs, the command provides two privileged requests that allow you to configure the values used by your site.

- Use `add_court_value` to add a new site court value.
- Use `remove_court_value` to remove an existing site court value.

All site court values you add or remove with these requests are completely local to your site and will be seen by the HUB only in audit text.

The default site court values provided with `site_call_system` are stored in the `site_call_system` program module. The first time you add or remove a site court value, the command creates the file `scs_configuration` in the site call database directory and puts into it a single record describing all of the site court values. Subsequently, when you reconfigure site court values, the command updates the file with the revised list.

Each time a user or process accesses the Site Call System database, `site_call_system` checks to see if `scs_configuration` exists and whether it has been updated. If so, it loads the values in the list into memory, ready for use.

See the description of `add_court_value` later in this command description for specifications on the site court value format.

Since `site_call_system` is embedded into both the `rsn_server.pm` and `site_call_xfer.pm` programs and may also run interactively, you must ensure that the processes running both those programs and all users of the Site Call System have appropriate access to the `site_call_system` configuration table file. (These files are not created until `site_call_system` is first executed and will inherit their access from the system directory, via the site call database directory.) If access is inappropriate, the processes and users may disagree over what is a valid site court value. The Explanation section earlier in this command description provides instructions on the required access.

Stratus advises that you do **not** change site court values frequently. Doing so increases the possibility of disagreement between different processes about the correct values. You also risk making the audit text in a call confusing: the call stores the array index of a site court value rather than the value itself, so the audit text may not reference the site court value that you intend.

Note that because modules run the Site Call System subsystem independently, several different versions may coexist on the same system. However, a Release 11 version **can** access calls in a pre-Release 11 site call database. To avoid compatibility problems between Release 11 and pre-Release 11 versions, `site_call_system` assigns values for the nonexistent site status and site court fields as follows:

- The value of the site status is the HUB status of the call.
- The value of the site court is the first site court value defined.

Use these values for reference when reading a call. The call itself is, however, unchanged until it is updated by the software.

To see a list of the current site court values, invoke the `list_court_values` request.

Site Call System Requests

This section describes all available Site Call System requests.

Once a call has been escalated to the HUB via the `site_call_system` command, a pair of asterisks are placed in the text to indicate this fact. For example:

```
** Call sent up to the HUB.
```

If you intend to use the text in another call (for example, using the `-output_path` option with the `display_call` request), you must edit the text to disable or remove the asterisks. Otherwise, the `site_call_system` command treats the text as if it already has been escalated in the current call and will not re-escalate it. This caution applies when using the site call form or any of the following requests.

```
add_call
add_court_value
change_call
display_call
help
list_calls
list_court_values
maint_request
new_call
purge_calls
quit
```

```
remove_court_value
rsn_mail
set_mode
update_calls
```

The `add_call` Request

The `add_call` request allows you to add a call to the site call database by specifying the call data as argument values either on the command line or in the display form. You can use this request to add a call to the site call database as a batch job.

To enter a call via the site call form, issue the `new_call` request.

Display Form

```
------------------------------------ add_call ----------------------------------
-summary:
-module:       current_module
-contact:      current_user
-phone:
-text:
-text_file:
-site_status: NEW                         -product:
-release:                                 -site_court:  SITE
-area:         SOFTWARE                   -seriousness: MINOR
-type:         QUESTION                   -escalate:    no
```

Command-Line Form

```
add_call -summary string
         [-module module_name]
         [-contact string]
         -phone string
         [-text string]
         [-text_file path_name]
         [-site_status string]
         [-product string]
         [-release string]
         [-site_court string]
         [-area string]
         [-seriousness string]
         [-type string]
         [-escalate string]
```

Arguments

- `-summary` *`string`* **Required**

  A character string that briefly describes the purpose of the call. You must supply this value.

- `-module` *`module_name`*

  The name of the module about which the call is made. If you do not provide a value, the name of the current module is used.

- `-contact` *`string`*

  The name of the person at your site designated as the main contact for the call. If you do not specify a name, the request uses the name of the current user.

- `-phone` *`string`* **Required**

  The phone number of the person specified in the `-contact` argument. You must supply a number.

- `-text` *`string`*

  A one-line character string to be added to the call's text buffer. If you specify both this argument and the `-text_file` argument, the text specified as the value of this argument is added to the call, followed by a blank line and the text from the specified file.

- `-text_file` *`path_name`*

  The path name of a file whose text is to be added to the call's text buffer. If you specify both this argument and the `-text` argument, the text specified as the value of the `-text` argument is added to the call, followed by a blank line and the text from the specified file.

- `-site_status` *`string`* CYCLE

  The site status of the call. Table 14-2 lists the values.

Table 14-2. `-site_status` Values of `add_call` Request

Value	Description
`CANCELLED`	Work on the issue has been abandoned, which indicates that the issue should not have been added in the first place; for example, it may have been a duplicate of some other issue.
`CLOSED`	Your company's issue has been resolved, or the answer to a question provided.
`LOCKED_CLOSED`	The issue is no longer active and will not be re-opened due to further updates your company provides.

Table 14-2. `-site_status` Values of `add_call` Request *(Continued)*

Value	Description
`NEW` (the default)	The issue was added recently and is not yet being worked on. The information in the **Issue Owner** field is not significant, and a support agent can take possession of the issue.
`OPEN`	Resolution of the issue is being pursued by a particular support agent.
`PENDING_CLOSE`	The issue is waiting for a system event to close it.
`REASSIGN`	The issue has been initially investigated and found to be assigned to the wrong area. It is awaiting pickup by another support agent.
`UNRESOLVED`	Work on the issue has terminated because a solution is not possible.

▶ `-product` *`string`*
A character string specifying the product about which you are submitting the call.

▶ `-release` *`string`*
A character string specifying the release of OpenVOS running on the module specified as the value of the `-module` argument. If you do not supply a value, the request uses the release of OpenVOS running on the current module.

▶ `-site_court` *`string`*
The person or group at your site responsible for the next action on the call. Issue the `list_court_values` request to see a list of allowed site court values. See “Configuring Site Court Values” earlier in this command description for information on how to add and remove site court values.

▶ `-area` *`string`* CYCLE
Character string specifying the subject of the call. Table 14-3 lists the values.

Table 14-3. `-area` Values of `add_call` Request

Value	Description
`DOCUMENT`	The issue is with Stratus documentation.
`HARD_RSN`	The issue resulted from a Remote Service Network (RSN) report.
`HARDWARE`	The issue is attributed to the system hardware.
`POWERFAIL`	The issue resulted from loss of power to the system.
`SOFTWARE` (the default)	The issue is attributed to the system software.

Table 14-3. `-area` Values of `add_call` Request *(Continued)*

Value	Description
`UNKNOWN`	The issue cannot be clearly attributed to any of the other areas.

▶ `-seriousness` *`string`* CYCLE
A character string specifying the seriousness of the call. Table 14-4 lists the values.

Table 14-4. `-seriousness` Values of `add_call` Request

Value	Description
`CRITICAL`	The issue has stopped or prevents continued production at your company. No avoidance procedure is available, and you need a resolution as soon as possible. Critical issues may or may not involve system crashes.
`MINOR` (the default)	The issue causes no particular problems for your company; it is incorrect behavior, but can be avoided or ignored.
`MODERATE`	The issue can be avoided or ignored, but causes continuing or recurring problems at a site.
`SERIOUS`	The issue has stopped development of your company's application. No reasonable avoidance procedure is available, and the issue prevents completion of development and subsequent production use of the system.

▶ `-type` *`string`* CYCLE
A character string identifying the purpose of the call. Table 14-5 lists the values.

Table 14-5. `-type` Values of `add_call` Request

Value	Description
`INSTALL`	The issue occurred during installation.
`ORDER`	The issue occurred while entering or processing an order.
`OTHER`	The issue does not fit into any of the categories above.
`PROBLEM`	The issue is a request to solve a problem.
`QUESTION` (the default)	The issue is a request for information.
`REBOOT`	The issue occurred during system reboot.

Table 14-5. `-type` Values of `add_call` Request *(Continued)*

Value	Description
`REQUEST`	The issue is a request for something other than information or a problem resolution.

▶ `-escalate` *`string`* CYCLE

Adds the call to the site call database and then sends it to the HUB (including both header and text). The default is `no`, which means the call is added to the site call database but not to the HUB.

The `add_court_value` Request (Privileged)

The `add_court_value` request adds a new value to the list of allowed site court values.

Display Form

```
------------------------------ add_court_value ------------------------------
new_court_value:
```

Command-Line Form

`add_court_value` *`new_court_value`*

Arguments

▶ *`new_court_value`* **Required**

The new site court value you want to add. It must be a character string containing from 1 to 10 characters. You can use the following:

- any uppercase or lowercase alphabetic characters
- any numeric characters
- the underline character (_)
- the dollar sign character (`$`)

The site court value cannot include any spaces.

You cannot add a value that either exactly duplicates an existing value or differs from an existing value only in the case of its alphabetic characters. For example, you cannot add the value `smith` to a list that already contains `SMITH` or `Smith`.

If you try to add an invalid or duplicate site court value, you receive an error message, and the configuration table file remains unchanged.

The `change_call` Request

The `change_call` request updates an existing call in the site call database. Once you issue the request with a specific call number, the site call form appears. The fields in the form are filled in with the current call's values, which you can then modify.

See the explanation of ‘‘The Site Call Form’’ earlier in this command description for more information.

Display Form

```
------------------------------- change_call --------------------------------
call_no:
-hub_call:
```

Command-Line Form

```
change_call [call_no]
            [-hub_call number]
```

Arguments

- ▶ `call_no`

  The call number of the local call you want to change. If you do not specify a value for *`call_no`*, either the most-recently modified call number appears, the field is left blank, or the call named by `-hub_call` appears. Unless you specify a HUB call number in the `-hub_call` argument, you must either specify a local call number explicitly, or use the default call number.

- ▶ `-hub_call` *`number`*

  The HUB call number associated with the local call you want to change. The `change_call` request retrieves the local call from the database. You must use the call number format given by the HUB servicing your system. See the description of the hybrid call format earlier in this command description for details.

  NOTE

  This argument overrides the `call_no` argument. If you specify a HUB call associated with a local call other than the current default call number, the default changes to the local call number corresponding to the HUB call.

The `display_call` Request

The `display_call` request displays one or more calls in the database according to the particular format you specify.

Display Form

```
------------------------------- display_call ------------------------------
call_no:
-hub_call:
-format:       full
-header:       yes
-output_path:
```

Command-Line Form

```
display_call [call_no ...]
            [-hub_call number ...]
            [-format string]
            [-no_header]
            [-output_path path_name]
```

Arguments

- ▶ *`call_no`*

  The call number(s) of one or more local calls you want to display. If you do not specify a value for *`call_no`*, the current default call number is used. Otherwise, the first call number specified becomes the default call number.

- ▶ `-hub_call` *`number`*

  The HUB call number(s) associated with one or more local calls you want to display. The `change_call` request retrieves the local call(s) from the database. You must use the call number format specified by the HUB servicing your system. See the description of the hybrid call format, earlier in this command description, for details.

  NOTE

  This argument overrides the *`call_no`* argument. If you specify a HUB call associated with a local call other than the current default call number, the default changes to the local call number corresponding to the HUB call.

▶ `-format` *`string`* CYCLE

A value specifying how the information about the specified calls is to appear. Possible values and their meanings follow.

- `full`

  The default; it displays the header and the complete text of each call.

- `headers_only`

  Displays only the information in the header section for each call.

- `1_line`

  Displays a one-line summary of each call under a line of title information.

- `1_line_trailer`

  Displays the one-line summary, and, following it, a table defining the codes appearing in each call's summary.

▶ `-no_header` CYCLE

Does not display the title information at the top of the one-line summaries appearing when you select either `1_line` or `1_line_trailer` as a value for the `-format` argument. This argument is meaningless if you specify either `full` or `headers_only` as a value for `-format`. The default is `yes`, which displays the header.

▶ `-output_path` *`path_name`*

The path name of a file to which the output of the `display_call` request is written.

See the note at the beginning of the section "Site Call System Requests" for information placed in the text once a call has been escalated to the HUB.

The `help` Request

The `help` request displays a list of available Site Call System requests.

Display Form

```
---------------------------------- help ---------------------------------
-match: █
```

Command-Line Form

`help -match` *`string`*

Arguments

- ► `-match` *`string`*
  A character string. If you specify a value for this argument, the `help` request displays only those Site Call System requests whose names contain this string. If you omit this argument, the `help` request displays all Site Call System requests.

The `list_calls` Request

The `list_calls` request displays all of the calls in the database that have the characteristics you specify, in one of several formats.

Display Form

```
-------------------------------- list_calls -------------------------------
output_path:       █
-format:           1_line
-status:
-site_status:
-contact:
-begin_date_time:
-end_date_time:
-updated_after:
-match:
-and_match:
-header:           yes                      -seriousness:
-type:                                      -area:
-module:                                    -release:
-product:                                   -site_court:
-first_call:                                -num_calls:
```

Command-Line Form

```
list_calls [output_path]
           [-format format_type]
           [-status string . . .]
           [-site_status string . . .]
           [-contact string]
           [-begin_date_time date_time]
           [-end_date_time date_time]
           [-updated_after date_time]
           [-match string]
           [-and_match string]
           [-no_header]
           [-seriousness string]
           [-type string]
           [-area string]
           [-module string]
           [-release string]
           [-product string]
           [-site_court string]
           [-first_call number]
           [-num_calls number]
```

Arguments

▶ `output_path`

The path name of a file to which the output of the `list_calls` request is written.

▶ `-format` *`format_type`* CYCLE

Specifies how the output should appear. Possible values and their meanings follow.

- `full`

  The default, which displays the header and the entire text of each call.

- `headers_only`

  Displays only the information in the header section for each call.

- `1_line`

  Displays a one-line summary of each call under a line of title information.

- `1_line_trailer`

  Displays the one-line summary and, following it, a table defining the codes appearing in each call's summary.

- `-status` *`string`*

  One or more values specifying a call's HUB status. This is not the same as the call's site status, which you specify in the `-site_status` argument. This value can be blank, or it can be one or more of the following: `NEW`, `OPEN`, `CLOSED`, `DELETED`, `UNRESOLVED`, `ACTIVE`, and `ALL`.

  If you leave this value blank, the current setting of the `omit_deleted` mode determines what calls you see. A value of `yes` lists calls whose HUB status is **not** deleted; a value of `no` lists all calls, regardless of HUB status. Use the `-omit_deleted` argument of either the `site_call_system` command or the `set_mode` request to set the value of the `omit_deleted` mode.

  If you specify `NEW`, `OPEN`, `CLOSED`, `DELETED`, or `UNRESOLVED`, you see the calls with that HUB status.

  If you specify `ACTIVE`, you see calls with a HUB status of `NEW` or `OPEN`.

  If you specify `ALL`, you see all calls, regardless of HUB status; you see all calls with HUB status of `NEW`, `OPEN`, `CLOSED`, `DELETED`, or `UNRESOLVED`.

  The HUB sets this form field value.

- `-site_status` *`string`*

  One or more values specifying a call's site status. This is not the same as the call's HUB status, which you specify in the `-status` argument. This value can be blank, or it can be one or more of the following: `NEW`, `OPEN`, `CLOSED`, `DELETED`, `UNRESOLVED`, `ACTIVE`, or `ALL`.

  If you leave this value blank, the current setting of the `omit_deleted` mode determines what calls you see. A value of `yes` lists calls whose site status is **not** deleted; a value of `no` lists all calls, regardless of site status. Use the `-omit_deleted` argument of either the `site_call_system` command or the `set_mode` request to set the value of the `omit_deleted` mode.

  If you specify `NEW`, `OPEN`, `CLOSED`, `DELETED`, or `UNRESOLVED`, you see the calls with that site status.

  If you specify `ACTIVE`, you see calls with a site status of `NEW` or `OPEN`.

  If you specify `ALL`, you see all calls, regardless of site status; you see all calls with site status of `NEW`, `OPEN`, `CLOSED`, `DELETED`, or `UNRESOLVED`.

  The site sets this value.

- `-contact` *`string`*

  A name specifying the person at your site designated as the main contact for a call. If you specify a value for *`string`*, it must match exactly the corresponding value in the `Contact` field in each call's database entry.

- `-begin_date_time` *`date_time`*

  The date (and, optionally, the time) on or after a call was entered into the database. The value specified as *`date_time`* must be in standard date/time format. The Site Call System determines this date and time from a call's `Time Added` field. If you specify the `-begin_date_time` argument, no calls entered earlier than the value of *`date_time`* are listed.

- `-end_date_time` *`date_time`*

  The date (and, optionally, the time) on or before a call was entered into the database. The value specified as *`date_time`* must be in standard date/time format. The Site Call System determines this date and time from a call's `Time Added` field. If you specify the `-end_date_time` argument, no calls entered later than the value of *`date_time`* are listed.

  The default time for this request is 0:00:00. If you specify the `-end_date_time` argument and specify only the date and not the time, calls entered on the specified date are **not** listed; to list all calls for a specific day, specify a date one day later or specify both the date and a time of 23:59:59.

- `-updated_after` *`date_time`*

  The date (and, optionally, the time) on or after a call was updated. The value specified as *`date_time`* must be in standard date/time format. The Site Call System determines this date and time from a call's `Updated` field. If you specify the `-updated_after` argument, no calls updated earlier than the value of *`date_time`* are listed.

- `-match` *`string`*

  A character string specifying calls in the database with a field in the call body containing the same characters, regardless of case.

- `-and_match` *`string`*

  A character string specifying calls in the database with both of the following (regardless of case):

  - a field with the same characters as those specified as the value for *`string`* in the `-match` argument
  - a field with the same characters as those specified as the value for *`string`* in the `-and_match` argument

  You cannot specify the `-and_match` argument unless you also specify the `-match` argument.

▶ `-no_header` CYCLE

Does not display the title information at the top of the one-line summaries appearing when you select either `1_line` or `1_line_trailer` as a value for the `-format` argument. This argument is meaningless if you specify either `full` or `headers_only` as a value for `-format`. The default is `yes`, which displays the header.

▶ `-seriousness` *`string`* CYCLE

A character string specifying a call's degree of seriousness. Available values for *`string`* are: `MINOR`, `MODERATE`, `SERIOUS`, and `CRITICAL`.

▶ `-type` *`string`* CYCLE

A character string specifying the purpose of a call. Available values for *`string`* are: `QUESTION`, `PROBLEM`, `REQUEST`, `OTHER`, `ORDER`, `INSTALL`, and `REBOOT`.

▶ `-area` *`string`* CYCLE

A character string specifying the subject of a call. Available values are: `SOFTWARE`, `HARDWARE`, `DOCUMENT`, `OTHER`, `HARD_RSN`, the blank character, or the null string (`''`). You can also omit the field entirely, to instruct `list_calls` not to match on this field.

▶ `-module` *`string`*

A character string specifying the name of the module about which a call was made. If you specify a value for *`string`*, it must match exactly the corresponding value in the `Module` field of a call's database entry.

▶ `-release` *`string`*

A character string specifying the release of OpenVOS running on the module about which a call was made. If you specify a value for *`string`*, it must match exactly the corresponding value in the `Release` field of a call's database entry.

▶ `-product` *`string`*

A character string specifying the name of the product about which a call was made. If you specify a value for *`string`*, it must match exactly the corresponding value in the `Product` field of a call's database entry.

▶ `-site_court` *`string`*

The person or group at your site responsible for the next action on the call. The default values for this argument are `HUB`, `SITE`, `ENGINEER`, and `USER`.

You can customize this list according to the needs of your site. See "Configuring Site Court Values" earlier in this command description for information on how to customize site court values. Issue the `list_court_values` request, documented later in this command description, to see a list of allowed site court values.

- `-first_call` *number*
  A number specifying the first call number in the database to be shown in the output of the `list_calls` request.

- `-num_calls` *number*
  A number specifying how many calls in the database are to be shown in the output of the `list_calls` request.

Examples

The `list_calls` request generates output with the following format.

```
call   hub       date    flags   court   module  summary
----   ---       ----    -----   -----   ------  -------

1      1234556   0101    ncshp   hub     m1      Test Call
```

The headings, with the exception of the `flags` heading, are self-explanatory. There are five single-position fields under the `flags` heading. Each position under `flags` corresponds to a field on the `SITE CALL SYSTEM` form.

The following table notes the position, the corresponding field on the `SITE CALL SYSTEM` form, the possible values that can appear in that position, and the character strings associated with these values.

Position	Field	Values	Character Strings

First	Site Status	`n`
		`o`
		`c`
		`d`
			`NEW`
			`OPEN`
			`CLOSED`
			`DELETED`

Second	Hub Status	`n`
		`o`
		`c`
		`d`
			`NEW`
			`OPEN`
			`CLOSED`
			`DELETED`

Third	Seriousness	`m`
		
		`s`
		`c`
			`MINOR`
			`MODERATE`
			`SERIOUS`
			`CRITICAL`

Position	Field	Values	Character Strings
Fourth	Area	`s`	`SOFTWARE`
		`h`	`HARDWARE`
		`d`	`DOCUMENT`
		`o`	`OTHER`
		`r`	`HARD_RSN`
		`s`	`SIMPLEXED`
		`p`	`POWERFAIL`
		`u`	`UNKNOWN`
			(blank)
Fifth	Type	`q`	`QUESTION`
		`p`	`PROBLEM`
		`r`	`REQUEST`
		`o`	`OTHER`
			`ORDER`
		`i`	`INSTALL`
		`r`	`REBOOT`

The flags in the example shown in the preceding Examples section provide the following information.

- The first field, with the value `n` for `NEW`, is the site status field. The value in this field changes only if someone at the site updates the call.
- The second field, with the value `c` for `CLOSED`, is the status field. The value in this field changes when someone at the CAC updates the call.
- The third field, with the value `s` for `SERIOUS`, is the seriousness field.
- The fourth field, with the value `h` for `HARDWARE`, is the area field.
- The fifth field, with the value `p` for `PROBLEM`, is the type field.

The `list_court_values` Request

The `list_court_values` request provides a list of the currently configured site court values and tells you how many more site court values you are allowed to configure. (You can configure up to 20 site court values.)

Display Form

```
----------------------------- list_court_values -----------------------------
No arguments required.  Press ENTER to continue. █
```

Command-Line Form

```
list_court_values
```

The `maint_request` Request

The `maint_request` request is incorporated into the Site Call System for convenience. It is slightly changed, however, from the OpenVOS command of the same name. For example, it does not allow either the `add_call` request or the `update_calls` request. Within the Site Call System, you can use the `new_call` request to add a call, and the `change_call` or `update_calls` requests to update a call. In addition, the `mail` request of the `maint_request` command is not available since `rsn_mail` is available as a Site Call System request.

For more information, see the description of the `maint_request` command in this chapter.

The `new_call` Request

The `new_call` request allows you to add a call to the site call database via the site call form. The site call form appears with all fields cleared, except those with default values.

Do not use the `new_call` request to add a call as a batch job. Instead, use the `add_call` request, which allows you to enter a call on the command line or via the display form.

Display Form

```
---------------------------------- new_call --------------------------------
No arguments required.  Press ENTER to continue. █
```

Command-Line Form

```
new_call
```

The `purge_calls` Request

The `purge_calls` request deletes from the database all calls that have the characteristics you specify, then returns a message stating the number of calls that have been deleted.

Display Form

```
-------------------------------- purge_calls --------------------------------
-status:
-site_status:
-contact:
-begin_date_time:
-end_date_time:
-updated_after:
-match:
-and_match:
-seriousness:                              -type:
-area:                                     -module:
-release:                                  -product:
-site_court                                -first_call:
-num_calls:                                -ask:          yes
```

Command-Line Form

```
purge_calls [-status string...]
            [-site_status string . . .]
            [-contact string]
            [-begin_date_time date_time]
            [-end_date_time date_time]
            [-updated_after date_time]
            [-match string]
            [-and_match string]
            [-seriousness string]
            [-type string]
            [-area string]
            [-module string]
            [-release string]
            [-product string]
            [-site_court string]
            [-first_call number]
            [-num_calls number]
            [-no_ask]
```

Arguments

- `-status` *string*

  One or more values specifying a call's HUB status. This is not the same as the call's site status, which you specify in the `-site_status` argument. This value can be blank, or it can be one or more of the following: `NEW`, `OPEN`, `CLOSED`, `DELETED`, `UNRESOLVED`, `ACTIVE`, and `ALL`.

If you leave this value blank, the current setting of the `omit_deleted` mode determines what calls you select for deletion. A value of `yes` selects calls whose HUB status is **not** deleted; a value of `no` selects all calls, regardless of HUB status. Use the `-omit_deleted` argument of either the `site_call_system` command or the `set_mode` request to set the value of the `omit_deleted` mode.

If you specify `NEW`, `OPEN`, `CLOSED`, `DELETED`, or `UNRESOLVED`, you delete calls with that HUB status.

If you specify `ACTIVE`, you delete calls with a HUB status of `NEW` or `OPEN`.

If you specify `ALL`, you delete all calls, regardless of HUB status; you delete all calls with the HUB status of `NEW`, `OPEN`, `CLOSED`, `DELETED`, or `UNRESOLVED`.

The HUB sets this form field value.

▶ `-site_status` *`string`*

One or more values specifying a call's site status. This is not the same as the call's HUB status, which you specify in the `-status` argument. This value can be blank, or it can be one or more of the following: `NEW`, `OPEN`, `CLOSED`, `DELETED`, `UNRESOLVED`, `ACTIVE`, or `ALL`.

If you leave this value blank, the current setting of the `omit_deleted` mode determines what calls you select for deletion. A value of `yes` selects calls whose site status is **not** deleted; a value of `no` selects all calls, regardless of site status. Use the `-omit_deleted` argument of either the `site_call_system` command or the `set_mode` request to set the value of the `omit_deleted` mode.

If you specify `NEW`, `OPEN`, `CLOSED`, `DELETED`, or `UNRESOLVED`, you delete calls with that site status.

If you specify `ACTIVE`, you delete calls with a site status of `NEW` or `OPEN`.

If you specify `ALL`, you delete all calls, regardless of site status; you delete all calls with the site status of `NEW`, `OPEN`, `CLOSED`, `DELETED`, or `UNRESOLVED`.

The site sets this value.

▶ `-contact` *`string`*

A name specifying the person at your site designated as the main contact for a call. If you specify a value for *`string`*, it must match exactly the corresponding value in the `Contact` field in each call's database entry.

▶ `-begin_date_time` *`date_time`*

The date (and, optionally, the time) on or after a call was entered into the database. The value specified as *`date_time`* must be in standard date/time format. The Site Call System determines this date and time from a call's `Time Added` field. If you specify the `-begin_date_time` argument, no calls entered earlier than the value of *`date_time`* are deleted.

▶ `-end_date_time` *`date_time`*

The date (and, optionally, the time) on or before a call was entered into the database. The value specified as *`date_time`* must be in standard date/time format. The Site Call System determines this date and time from a call's `Time Added` field. If you specify the `-end_date_time` argument, no calls entered later than the value of *`date_time`* are deleted.

▶ `-updated_after` *`date_time`*

The date (and, optionally, the time) on or after a call was updated. The value specified as *`date_time`* must be in standard date/time format. The Site Call System determines this date and time from a call's `Updated` field. If you specify the `-updated_after` argument, no calls updated earlier than the value of *`date_time`* are deleted.

▶ `-match` *`string`*

A character string specifying calls in the database with a field containing the same characters, regardless of case.

▶ `-and_match` *`string`*

A character string specifying calls in the database with both of the following (regardless of case):

- a field with the same characters as those specified as the value for *`string`* in the `-match` argument
- a field with the same characters as those specified as the value for *`string`* in the `-and_match` argument—You cannot specify the `-and_match` argument unless you also specify the `-match` argument.

▶ `-seriousness` *`string`* CYCLE

A character string specifying a call's degree of seriousness. Available values for *`string`* are: `MINOR`, `MODERATE`, `SERIOUS`, and `CRITICAL`.

▶ `-type` *`string`* CYCLE

A character string specifying the purpose of a call. Available values for *`string`* are: `QUESTION`, `PROBLEM`, `REQUEST`, `OTHER`, `ORDER`, `INSTALL`, and `REBOOT`.

▶ `-area` *`string`* CYCLE

A character string specifying the subject of a call. Available values are: `SOFTWARE`, `HARDWARE`, `DOCUMENT`, `OTHER`, `HARD_RSN`, the blank character, or the null string (`''`). You can also omit the field entirely, to instruct `purge_calls` not to match on this field.

- ▶ `-module` *`string`*

  A character string specifying the name of the module about which a call was made. If you specify a value for *`string`*, it must match exactly the corresponding value in the `Module` field of a call's database entry.

- ▶ `-release` *`string`*

  A character string specifying the release of OpenVOS running on the module about which a call was made. If you specify a value for *`string`*, it must match exactly the corresponding value in the `Release` field of a call's database entry.

- ▶ `-product` *`string`*

  A character string specifying the name of the product about which a call was made. If you specify a value for *`string`*, it must match exactly the corresponding value in the `Product` field of a call's database entry.

- ▶ `-site_court` *`string`*

  The person or group at your site responsible for the next action on the call. Issue the `list_court_values` request, documented previously in this command description, to see a list of allowed site court values.

  See "Configuring Site Court Values" earlier in this command description for information on how to customize site court values.

- ▶ `-first_call` *`number`*

  A number specifying the first call number in the database to be deleted by the `purge_calls` request.

- ▶ `-num_calls` *`number`*

  A number specifying how many calls in the database are to be deleted by the `purge_calls` request.

- ▶ `-no_ask` CYCLE

  Does not prompt you before deleting the specified calls from the database. The default is `yes`, meaning `purge_calls` prompts you for confirmation.

The `quit` Request

The `quit` request exits the Site Call System and returns your process to command level. There is no display form for this request.

Command-Line Form

```
quit
```

The `remove_court_value` Request (Privileged)

The `remove_court_value` request removes an existing site court value by deleting it from the file `scs_configuration` in the site call directory.

Display Form

```
---------------------------- remove_court_value ---------------------------
court_value: court_value
```

Command-Line Form

```
remove_court_value court_value
```

Arguments

▶ *`court_value`* CYCLE **Required**

The site court value you want to remove. You cannot remove a site court value to which the site court field for any call in the site call database is currently set. The default is the first site court value in the `scs_configuration` file.

The `rsn_mail` Request

The `rsn_mail` request is incorporated into the Site Call System for convenience. It is slightly changed, however, from the OpenVOS command of the same name. For example, it does not allow either the `Add an RSN Call` request or the `Update an RSN Call` request. Within the Site Call System, you can use the `new_call` request to add a call, and the `change_call` or `update_calls` request to update a call.

See the `rsn_mail` command for more information.

The `set_mode` Request

You use the `set_mode` request to either display or change the current settings of `site_call_system` modes. To display the current settings, invoke the request without specifying a mode or setting. To change current settings, specify a mode and setting.

The modes whose settings you can change are named `omit_deleted` and `show_audit`.

- Use the `omit_deleted` mode to specify whether the `list_calls` request displays deleted calls when you specify no HUB or status value. If `-omit_deleted` is enabled, no deleted calls are listed. If `-no_omit_deleted` is enabled, deleted calls are listed. For more information, see the `list_calls` request, and the `-status` and `-site_status` argument descriptions.
- Use the `show_audit` mode to specify whether the `update_calls` request displays audit text. If `-show_force` is enabled, audit text is displayed. If `-no_show_force` is enabled, audit text is not displayed. See the `update_calls` request description for more information.

Display Form

```
---------------------------------- set_mode --------------------------------
mode:     █
setting: on
```

Command-Line Form

```
set_mode [mode]
         [setting]
```

Arguments

▶ *mode* CYCLE

The name of the `site_call_system` mode whose setting is to be changed. The allowed values are `omit_deleted` and `show_audit`. If you omit this argument, the request displays the current values of all modes.

▶ *setting* CYCLE

The setting of the specified mode. The allowed values are `on` and `off`. If you specify a mode and omit this argument, the mode is set to `on` (the default). If you do not specify a value for the *mode* argument, this argument is ignored.

The `update_calls` Request

The `update_calls` request allows you to update one or more fields, in one or more calls, in the database with a single request. Each value specified using arguments to the `update_calls` request replaces the current value in the corresponding field of each call specified.

Note that when you issue the `update_calls` request, any fields with blank values are ignored.

You can choose to have this request display the audit text it generates when you update a call by setting the value of the `show_audit` mode to `on` before performing the update. Use the `-show_audit` argument of the `site_call_system` command or the `set_mode` request to set the value of the `show_audit` mode.

NOTE

See the note at the beginning of the section “Site Call System Requests” for information placed in the text once a call has been escalated to the HUB.

Display Form

```
---------------------------------- update_calls ------------------------------
call_no:
-summary:
-module:
-contact:
-phone:
-text:
-text_file:
-update_type: local                          -site_status:
-product:                                    -release:
-site_court:                                 -area:
-seriousness:                                -type:
```

Command-Line Form

```
update_calls call_no
             [-summary string]
             [-module module_name]
             [-contact string]
             [-phone string]
             [-text string]
             [-text_file path_name]
             [-update_type string]
             [-site_status string]
             [-product string]
             [-release string]
             [-site_court string]
             [-area string]
             [-seriousness string]
             [-type string]
```

Arguments

▶ `call_no` **Required**

The call number(s) of one or more calls in the database to be updated with values specified in any of the other arguments to the `update_calls` command. If you do not specify a value for *`call_no`*, the default call number is used.

▶ `-summary` *`string`*

A character string that briefly describes the purpose of the call. This value replaces the current value in the `Summary` field of the database entry for each call specified in the *`call_no`* argument.

▶ `-module` *`module_name`*

The name of the module about which a call is made. This value replaces the current value in the `Module` field of the database entry for each call specified in the *`call_no`* argument.

▶ `-contact` *`string`*

The name of the person at your site designated as the main contact for a call. This value replaces the current value in the `Contact` field of the database entry for each call specified in the *`call_no`* argument.

▶ `-phone` *`string`*

The phone number of the person specified in the `-contact` argument. This value replaces the current value in the `Phone` field of the database entry for each call specified in the *`call_no`* argument.

- ▶ `-text` *`string`*

  A one-line character string to be added to the text buffer of each call specified in the *`call_no`* argument. If you specify both this argument and the `-text_file` argument, the text specified as the value of this argument is added to the call, followed by a blank line and the text from the specified file.

- ▶ `-text_file` *`path_name`*

  The path name of a file whose text is to be added to the text buffer of each call specified in the *`call_no`* argument. If you specify both this argument and the `-text` argument, the text specified as the value of the `-text` argument is added to the call, followed by a blank line and the text from the specified file.

- ▶ `-update_type` *`string`* CYCLE

  A character string specifying the type of update to be performed on a call when you issue the `update_calls` request. Available values are: `local` (the default), `hub`, or `extended_hub`. These values correspond to the three update action keys available in the site call form: `Local Update`, `Partial Escalation`, and `Extended Escalate`, respectively. Their meanings are as follows:

  - `local`

    Updates the call into the local database only.

  - `hub`

    Updates the call into the local database. Then, the fields in the header section and only the text in the current update are sent to the HUB via the RSN. (Any text that was part of a previous local update is not sent to the HUB.) Text from previous local calls can be sent to the HUB only if it has been re-added to the call.

  - `extended_hub`

    Updates the call into the local database. Then, along with the header and the text in the current update, it sends any text added since the last escalation (or `hub` update). If there was no previous escalation, it sends all text from the beginning of the text buffer (which includes any previous local updates).

  Any values replaced or added with arguments to the latest `update_calls` request are included in the current update of calls specified in the *`call_no`* argument.

- ▶ `-site_status` *`string`* CYCLE

  The site status of the call. This is not the same as the HUB status. This value can be blank, or it can be one of the following: `NEW`, `OPEN`, `CLOSED`, `DELETED`, or `UNRESOLVED`. If you do not specify a value, the command uses the current HUB status.

- ▶ `-product` *`string`*

  A character string specifying the product about which you are submitting the call.

- ▶ `-release` *`string`*

  A character string specifying the release of OpenVOS running on the module about which a call was made. This value replaces the current value in the `Release` field of the database entry for each call specified in the *`call_no`* argument.

- ▶ `-site_court` *`string`*

  The person or group at your site responsible for the next action on the call. To see a list of allowed site court values, issue the `list_court_values` request, documented previously in this command description.

  For information on how to add and remove site court values, see "Configuring Site Court Values" earlier in this command description.

- ▶ `-area` *`string`* CYCLE

  A character string specifying the subject of the call. This value can be blank or one of the following: `SOFTWARE`, `HARDWARE`, `DOCUMENT`, `OTHER`, `SIMPLEXED_HW`, `POWERFAIL`, `UNKNOWN`, `HARD_RSN`, or the null string (`''`). This value replaces the current value in the `Area` field of the database entry for each call specified in the *`call_no`* argument. You can also omit the field entirely, to instruct `update_calls` not to change the current values in this field.

- ▶ `-seriousness` *`string`* CYCLE

  A character string specifying a call's degree of seriousness. This value can be either blank or one of the following: `MINOR`, `MODERATE`, `SERIOUS`, or `CRITICAL`. This value replaces the current value in the `Seriousness` field of the database entry for each call specified in the *`call_no`* argument.

- ▶ `-type` *`string`* CYCLE

  A character string identifying the purpose of the call. This value can be either blank or one of the following: `QUESTION`, `PROBLEM`, `REQUEST`, `OTHER`, `ORDER`, `INSTALL`, or `REBOOT`. This value replaces the current value in the `Type` field of the database entry for each call specified in the *`call_no`* argument.

start_rsn

Purpose

The `start_rsn` command is a system process command. It is invoked **only** from the `mddmon_start_up.cm` file, which, in turn, is started by the `start_stcp.cm` file.

The `start_rsn` command starts processes that are used for the RSN, thus enabling the RSN on a module. If the file required by the Site Call System subsystem exists on the specified bridge module, this command also starts the SiteCallXfer process that manages calls coming from the HUB to the site database.

NOTE

If you are using the RSN over the X.25 communications protocol, see the manual *VOS Communications Software: X.25 and StrataNET Administration* (R091).

Display Form

```
----------------------------------- start_rsn --------------------------------
site_id:
out_channel_name:
in_channel_name:
phone_number:
-bridge_module:    current_bridge_module
-dialer_type:      console_server.modem
-log:              yes
-module:           current_module
-backup_phone:
-use_pending:      no
-recreate_table:   no
-truncate_queue:   no
```

Command-Line Form

```
start_rsn site_id
          out_channel_name
          in_channel_name
          phone_number
          [-bridge_module bridge_module_name]
          [-dialer_type name]
          [-no_log]
          [-module module_name]
          [-backup_phone backup_phone_number]
          [-use_pending]
          [-recreate_table]
          [-truncate_queue]
```

Arguments

▶ *`site_id`* **Required**

The identifier assigned to your site by Stratus; it is not necessarily the same as the system name. This identifier cannot include the characters `%`, `<`, `>`, `!`, or `#`.

> NOTE
>
> Before you issue the `start_rsn` command, you **must** coordinate any changes in the value of the *`site_id`* argument and the value of the *`site_id`* argument of the `update_rsnip_site` command with the CAC or HUB because the *`site_id`* values of these two commands and the *`site_id`* value stored at the HUB must all be identical.

▶ *`out_channel_name`* **Required**

The name of the outgoing RSN slave device. This name is assigned to the outgoing RSN slave device in the `devices.table` file.

▶ *`in_channel_name`* **Required**

The name of the incoming RSN slave device. This name is assigned to the incoming RSN slave device in the `devices.table` file.

▶ *`phone_number`* **Required**

The meaning of this argument and the value that you specify for it depends upon the type of console server, as follows:

- When a system is configured with an RSN Internet console server, the value is ignored.

- When a system is configured with an RSN dialup console server (that is, a console server attached to a modem), *`phone_number`* specifies the primary phone number that the autodial modem calls when the site needs to transmit information to the HUB, such as the occurrence of hardware errors. Stratus assigns this number when your system is configured with an RSN dialup console server. The number must include all digits necessary to dial the number using the phone line to which the modem is connected. For example, if the assigned number is 800-555-1234, the value specified by *`phone_number`* would be `18005551234`. Note that only digits are allowed in *`phone_number`*.

▶ `-bridge_module` *`bridge_module_name`*
Specifies the name of the module to which the console server is connected. On the bridge module, `start_rsn` starts two processes, MaintServer and MaintBridgeServer. If the `site_call_db` file exists, `start_rsn` also starts the SiteCallXfer process. On other modules, the command starts only the MaintServer process.

If you do not specify this argument, OpenVOS assumes that the current module is the bridge module. To ensure consistency, you should specify the bridge module in the `start_rsn` command on every module in a multimodule system. This argument is optional only on a single-module system.

▶ `-dialer_type` *`name`*
Specifies the type of modem that the console server uses for the RSN connection. The value you specify depends on the type of console server, as follows:

- When a system is configured with an RSN Internet console server, specify `(master_disk)>system>null.modem`.
- When a system is configured with an RSN dialup console server (that is, a console server attached to a modem), specify `(master_disk)>system>moxa.modem`.

▶ `-no_log` CYCLE
Ceases to log RSN interactions to the `remote_maint_log.`*`date`* file on the bridge module. The default is `yes`, which logs the interactions.

▶ `-module` *`module_name`*
Specifies the module on which the RSN processes are to be started. The default is the current module. Under most circumstances, this default should not be changed.

- `-backup_phone` *`backup_phone_number`*

  The meaning of this argument and the value that you specify for it depends upon the type of console server, as follows:

  - When a system is configured with an RSN Internet console server, the value is ignored.
  - When a system is configured with an RSN dialup console server (that is, a console server attached to a modem), *`phone_number`* specifies the phone number that the autodial modem calls when the site needs to transmit information to the HUB, such as the occurrence of hardware errors, and when *`phone_number`* is busy or not in service. Stratus assigns the backup phone number, if available. The value specified by *`backup_phone_number`* must include all digits necessary to dial the number using the phone line to which the modem is connected. For example, if the assigned number is 800-555-6789, the value specified by *`backup_phone_number`* would be `18005556789`. Note that only digits are allowed in *`backup_phone_number`*.

- `-use_pending` CYCLE

  Discards any messages or requests already in the queue. By default (`no`), the command does not discard any messages or requests already in the queue.

- `-recreate_table` CYCLE

  Forces the `rsn_bridge_server.table` table to be re-created. By default (`no`), the command does not re-create this table.

- `-truncate_queue` CYCLE

  Truncates the `HUB.comm_MQ` message queue. By default (`no`), the command does not truncate this queue.

Explanation

Include this command in the `mddmon_start_up.cm` file (which is started by the `start_stcp.cm` file) on each module of a system that is part of the RSN. Place it after the `start_process` command that starts the Overseer process. On a multimodule system, the `start_rsn` command should be the same in each module's startup file.

This command starts the SiteCallXfer process only if the `(master_disk)>system>site_call_system>site_call_db` file exists on the bridge module specified in the command. It does so by invoking the `start_site_call_xfer` command.

When a system is configured with an RSN dialup console server (that is, a console server attached to a modem) and a backup number is available, be sure to include the `-backup_phone` argument in the `start_rsn` command line in `mddmon_start_up.cm`. Then, if the primary phone number is busy or not in service when communication is required, the MaintBridgeServer process tries the backup phone number. If the backup is also unavailable, the MaintBridgeServer process alternates between the primary and backup phone numbers, at intervals, until it completes a connection.

NOTE

The phone line used by the autodial modem to make these calls must be a dedicated outside line used only to connect to the CAC.

The `devices.table` file contains two entries (*`out_channel_name`* and *`in_channel_name`*) for the MaintBridgeServer process that are no longer needed for current releases of OpenVOS.

Examples

The following command starts processes that are used for the RSN on the current module at the site `ABC`, and the current module is configured with an RSN Internet console server. The outgoing TELNET slave device for connections to the console server device is `rsn_out.m1` on module `m1`. Since this RSN connection uses the Internet, the dialer type is the null-modem file. The incoming TELNET slave device for connections from the console server device is `rsn_in.m1` on module `m1`.

```
start_rsn ABC rsn_out.m1 rsn_in.m1 -bridge_module %s1#m1
           -dialer_type %s1#m1_mas>system>null.modem
```

The following command starts processes that are used for the RSN on the current module at the site `XYZ`, and the current module is configured with a RSN dialup console server. The outgoing TELNET slave device for connections to the console server device is `rsn_out.m1` on module `m1`, and the modem calls the phone number 1-800-555-1234. The backup phone number is 1-800-555-6789. The incoming TELNET slave device for connections from the console server device is `rsn_in.m1` on module `m1`.

```
   start_rsn XYZ rsn_out.m1 rsn_in.m1 18005551234
   -bridge_module m1 -backup_phone 18005556789
```

If the `(master_disk)>system>site_call_system>site_call_db` file exists on the bridge module, this command also starts the SiteCallXfer process.

Related Information

For more information on the RSN, see Chapter 6 as well as the descriptions of the related commands `rsn_mail_janitor`, `site_call_system`, and `update_rsnip_site`. See also the following manuals:

- the *OpenVOS STREAMS TCP/IP Administrator's Guide* (R419) for information on the `start_stcp.cm` file
- the manual *Migrating VOS Applications from Continuum Systems* (R607) for information on the `mddmon_start_up.cm` file
- the manual *OpenVOS System Administration: Configuring a System* (R287) for information about configuring OpenVOS for the RSN
- the site planning guide as well as the operation and maintenance guide for your system (as listed in Related Manuals in the Preface) for information about physically connecting the console server to the RSN bridge module

start_site_call_xfer

Purpose

The `start_site_call_xfer` command starts the SiteCallXfer process. This process executes the `site_call_xfer.pm` program, which manages call updates coming from the HUB to the local site call database.

Display Form

```
----------------------------- start_site_call_xfer ---------------------------
bridge_module:
```

Command-Line Form

`start_site_call_xfer` *`bridge_module`*

Arguments

- ▶ *`bridge_module`* **Required**

  The name of the bridge module on which the SiteCallXfer process is to be started.

Explanation

The `start_site_call_xfer` command starts a `SiteCallXfer` process that executes the privileged `site_call_xfer.pm` program. This program receives call updates from the HUB and then posts them to the local site call database.

This command is normally executed by the user only once, immediately after the site call database and database files are created by the first execution of `site_call_system`.

Subsequently, the SiteCallXfer process is initiated by the `start_rsn` command, when `module_start_up.cm` is run at module reboot.

Related Information

See the command descriptions of `site_call_system` and `start_rsn` in this chapter.

`tp_overseer` ***Privileged***

Purpose

This is a system process command. It is used **only** within `module_start_up.cm` files.

The `tp_overseer` command causes the executing process to operate as a transaction processing Overseer (TPOverseer). It sets some of the parameters for transaction processing on the module and also enables you to recover from a duplexed disk failure.

Display Form

```
----------------------------------- tp_overseer ------------------------------
-tlf_size:                10000
-twa_size:                5000
-twa_free_percent:        50
-twa_speed_txns:          5000
-protect_transactions:    yes
-keep_transaction_log:    no
-logical_recovery_only:   no
-metering:                extended
-max_metering:            extended
```

Command-Line Form

```
tp_overseer [-tlf_size number]
            [-twa_size number]
            -twa_free_percent number
            -twa_speed_txns number
            [-no_protect_transactions]
            [-keep_transaction_log]
            [-logical_recovery_only]
            [-metering]
            [-max_metering]
```

Arguments

- ▶ `-tlf_size` *`number`*

  Specifies the number of blocks to allocate to each transaction log file created. The default is `10,000` blocks.

- ▶ `-twa_size` *`number`*

  Specifies the number of blocks to allocate to the transaction work area (TWA), which is a file that the TPOverseer uses as a work area when committing a transaction. The TPOverseer stores modified copies of transaction-protected blocks in the TWA during processing. The default is `5,000` blocks. See TWA Arguments for more information.

- ▶ `-twa_free_percent` *`percentage`* **Required**

  Specifies the percentage of free pages in the TWA that are necessary to prevent a speed flush. Specify a value `20` through `50` (the default), inclusive. See TWA Arguments for more information.

- ▶ `-twa_speed_txns` *`number`* **Required**

  Specifies the number of concurrent transactions that have been committed but not yet applied to the transaction-protected files. The number of committed transactions determines when the TPOverseer selects a speed flush. For performance, concurrent transactions are committed, even if they are not yet applied to transaction-protected files, in order to reduce the number of disk I/Os to busy files This argument allows you to reduce the number of these disk reads and writes. Specify a value `1000` or larger. By default, the value is `5000`. To maximize performance, the value should always be `5000` or larger. See TWA Arguments for more information.

- ▶ `-no_protect_transactions` CYCLE

  Does not protect transactions against module interruption. The default flushes the transaction log files to disk. When you specify the `-no_protect_transactions` argument, transactions are processed more quickly, but they are not protected against module interruption.

  Regardless of whether you specify `-no_protect_transactions`, the TPOverseer reads the `(master_disk)>system>transaction_logs>tlf*` log during initialization to determine the state of the transactions that were previously running on the module. An orderly shutdown of the TPOverseer always produces logs indicating that all transactions are resolved. Module interruptions, however, often leave unresolved transactions.

  When you restart the TPOverseer after a module interruption where the TPOverseer was running with `-no_protect_transactions`, the TPOverseer looks at the `system>transaction_logs>tlf*` log to determine the state of the transactions previously running on the module. The TPOverseer will not start if the `system>transaction_logs>tlf*` log indicates that unresolved transactions exist. In this case, delete the `system>transaction_logs>tlf*` log before specifying the `tp_overseer` command.

When you invoke `tp_overseer` without the `-no_protect_transactions` argument, **do not** delete the `system>transaction_logs>tlf*` log. If you delete the log, the TPOverseer cannot finish any transactions that were started before the module failed.

▶ `-keep_transaction_log` CYCLE
Deletes old transaction logs when they are no longer needed. To use the `tp_restore` command with transaction logs, this argument must be set to `yes`. The default value is `no`.

▶ `-logical_recovery_only` CYCLE
Assists in the recovery from a duplexed disk failure and should **not** be used except in extreme circumstances. The default is `no`, meaning OpenVOS is not prepared to recover from a duplexed disk failure.

▶ `-metering` CYCLE
Specifies the current metering setting. Possible settings are `none`, `basic`, `queue`, and *`extended`*. The following list describes these metering settings.

- The *`none`* setting indicates that metering is not enabled.
- The `basic` setting enables *basic metering*, which involves counter-type meters; maintaining these meters generally has little effect on performance.
- The *`queue`* setting enables *`queue metering`*, which gathers queue statistics for actual and pseudo queues. (*`Pseudo queues`* provide a view of a transaction that often corresponds more closely to a user's perspective than does an actual queue.) The overhead of performing queue metering has some effect on performance because it involves reading the system clock; therefore, you may want to disable queue metering in performance-critical situations.
- The *`extended`* setting enables *`extended metering`*, which records the maximum values of queue meters and other per-transaction values over a given time interval. You can then display this information by using the `transaction_meters` request of the `analyze_system` command. Extended metering requires no additional clock reading but does require additional space in the Transaction State Info (TSI) structure. (The TSI structure, which exists from the time at which a transaction is started until all file records modified within the transaction physically exist on disk, contains information about the transaction and its current state.) Once you specify the maximum value, you cannot specify a greater value unless you restart the TPOverseer.

The default value is *`extended`*.

▶ `-max_metering` CYCLE

Specifies the maximum allowable metering setting that you can specify without restarting the TPOverseer. Possible settings are `queue` and *`extended`*, each of which is described in the *`-metering`* argument description. The default value is *`extended`*.

Explanation

The `tp_overseer` command uses the OpenVOS symbol table, which is created each time a new version of OpenVOS is booted. If the `.out` file of the TPOverseer process (`tp_overseer.out`) contains an error message related to the symbol table file, you must create a new symbol table using the `create_os_symtab` command.

In its normal processing, the TPOverseer updates the transaction work area, and then transfers the updates to particular transaction files. When a disk recovers, the blocks on the disk are physically recovered. During *physical recovery*, if the logs indicate the updates have been made to the transaction work area, but not transferred to the proper transaction files, the TPOverseer copies the proper pages of the work area to the correct files, using records in the logs that show which pages of the work area correspond to various pages of the transaction files.

However, after a duplexed disk failure, it is not always possible to recover the damaged disk to the exact physical state it was in when the failure occurred. As a result, physical recovery by the TPOverseer cannot be guaranteed for a duplexed disk failure.

To recover from duplexed disk failure, perform the following steps.

1. Execute `dump_transaction_log` on the current transaction logs to determine which files on the module with the damaged disk have unfinished transactions to be reapplied.

   NOTE

   If there are no unfinished transactions to be reapplied, or there are no `TLF_PHYSICAL_COMMIT` records, you can restart the TPOverseer normally when the disk is repaired or replaced and the files have been re-created. Make a note of **all** files for which a `TLF_PHYSICAL_COMMIT` record exists, even if the file is located on a disk other than the damaged disk.

2. Repair or replace the damaged disk.
3. Copy the transaction logs from `(master_disk)>system>transaction_logs` to a restore directory.
4. Start the TPOverseer.

5. Restore the files and the transaction logs identified by the `dump_transaction_log` command. Restore the files to their original directories, and the transaction logs to the `restore` directory.
6. Execute `tp_restore` on the restored files to roll the files forward and create versions logically equivalent to the files that existed before the disk failure.

   NOTE

   Do the same for **all** files found during Step 1 for which a `TLF_PHYSICAL_COMMIT` record existed, even if it resided on a disk other than the one damaged, since TPOverseer recovery cannot be performed on a selective basis. You cannot specify which files will be recovered.

7. Stop the `tp_overseer` command.

   NOTE

   If applications were inadvertently started during these steps, rerun `dump_transaction_log` to see which transaction files were modified.

8. Copy the transaction logs back to the `(master_disk)>system>transaction_logs` directory, and the transaction files back to their appropriate directories.
9. Invoke `tp_overseer` with the `-logical_recovery_only` argument. This causes all transaction updates to recur logically rather than physically.

   NOTICE

   If you invoke the `tp_overseer -logical_recovery_only` command without having performed the preceding steps, files may be corrupted.

10. Restart the applications.

In this situation, if there were transactions for which a `TLF_PHYSICAL_COMMIT` record was written, but no `TLF_PHYSICAL_DONE` record had been written at the time of the interruption, only some of a transaction's updated pages may have been dumped to disk. The logical reapplication of updates to the file could then lead to corruption. For example, the index may contain extra entries or multiple entries of a deleted record.

Consequently, you should specify the `-logical_recovery_only` argument only when there is a damaged disk, or if, for some reason, you have performed Steps 1 and 2 in the preceding list for **all** files on the modules for which a `TLF_PHYSICAL_COMMIT` record exists.

TWA Arguments

The `-twa_free_percent` and `-twa_speed_txns` arguments allow you to adjust the thresholds that the TPOverseer uses to determine when it needs to perform a speed flush. The *TWA* is a file that the TPOverseer uses as a work area when committing a transaction. In a *speed flush*, the TPOverseer moves modified blocks from the TWA into the transaction-protected files immediately, rather than waiting for a period of time. Although a speed flush is more disk-intensive and less efficient than the default waiting behavior, the data is always accurate. After a speed flush occurs, the default waiting behavior is automatically effective until the next time that these thresholds are reached.

You can use the following formula to calculate the speed-flush threshold:

`percentage` * *`twa_size`* = *`speed_flush_threshold`*

For example, assume that `-twa_free_percent` is set to 50, and `-twa_size` is set to 500 (the default). The *`percentage`* (50%) of *`twa_size`* (500 blocks) is equal to *`speed_flush_threshold`* (250 blocks). Therefore, for a 500-block file, a speed flush occurs when fewer than 250 blocks are free.

You should change the values of the `-twa_free_percent` and `-twa_speed_txns` arguments only if the following message appears in the `.out` file of the TPOverseer process (`tp_overseer.out`) or in the `syserr_log.(date)` file:

```
TWA too small, Speed flush selected
```

If you choose to ignore the message, the TPOverseer continues to work correctly. However, if you adjust the values as described in this section, you may notice a moderate improvement in performance.

Changing the value of `-twa_speed_txns` is effective only if your module processes over 500 transactions per second. In this case, you may notice a performance gain if you increase this value above 5000.

If the TWA size is smaller than 2001 blocks, the value of `-twa_free_percent` is 50, regardless of what you specify in *`percentage`*.

If the TWA size is in the range 2001 to 4999 blocks, *`speed_flush_threshold`* is at least 1000 blocks, regardless of the value you specify in *`percentage`*. This value prevents the number of free pages in the TWA from reaching zero, which would result in the brief suspension of transaction processing while the TPOverseer writes modified blocks in the TWA to the transaction-protected files.

If the TWA size is 5000 or more blocks, the TPOverseer uses the value you specify in `-twa_free_percent`. For very large TWA files (5000 to 3200 blocks), less than 50% of the TWA needs to be free to sustain adequate performance.

Examples

The following example shows the procedure you would follow to recover files on a module with a damaged disk. Module `s1` has two disks, `d01` and `d02`. Disk `d02` is damaged. Files `sales.feb` and `sales.mar` reside on disk `d01`, and files `reports.jun` and `reports.sep` reside on disk `d02`. To recover the files, perform the following steps.

1. Issue the command `dump_transaction_log` to determine what `TLF_PHYSICAL_COMMIT` records exist. Table 14-6 lists the types of recovery required for each file.

Table 14-6. Types of Recovery

Files	**`TLF_PHYSICAL_COMMIT` Record Exists**	**Type of Recovery Required**
No files	Yes	Physical (normal) recovery
`sales.feb` or `sales.mar` only	Yes	Physical (normal) recovery
`reports.jun` and/or `reports.sep` only	Yes	Logical recovery
(`sales.feb` or `sales.mar`) and (`reports.jun` or `reports.sep`)†	Yes	Logical recovery

† Any mixture of files from both disks.

Perform Steps 2 and 3 for all files with `TLF_PHYSICAL_COMMIT` records, where logical recovery is required.

2. Restore files from the previous save and invoke `tp_restore` to roll the files forward to the latest complete transactions.
3. Restart the TPOverseer with the `-logical_recovery_only` argument.

The following example shows the `tp_overseer` command as it might appear in a `module_start_up.cm` file.

```
start_process tp_overseer -priority 9 -privileged
     -process_name TPOverseer -output_path tp_overseer.out
```

Related Information

See the manual *OpenVOS System Administration: Starting Up and Shutting Down a Module or System* (R282) and the `create_os_symtab` command description in this chapter. See also the OpenVOS Transaction Processing Facility Reference manuals and the `module_start_up.cm` file shipped with the installation software. This file is stored in the directory `(master_disk)>system>release_dir`. For more information on the `tp_restore` command, see the *VOS Transaction Processing Facility Guide* (R215).

unload_kernel_program ***Privileged***

Purpose

The `unload_kernel_program` command unloads a program module that is currently loaded into the OpenVOS kernel.

Display Form

```
--------------------------- unload_kernel_program ---------------------------
kernel_program:
```

Command-Line Form

`load_kernel_program` *`kernel_program`*

Arguments

▶ *`kernel_program`* **Required**

The path name of the program module to be unloaded from the kernel.

Explanation

The `unload_kernel_program` command unloads a specified program module (for example, an I/O driver or a user-routine library) that is currently loaded into the OpenVOS kernel.

Kernel-loadable program modules are produced by the `bind` command. The `load_kernel_program` command loads a copy of the program module into the kernel address space.

Examples

The following command unloads the program module `stcp.cp.pm`, which is loaded on the current module in the current directory. Note that the `.pm` suffix is optional.

```
unload_kernel_program stcp.cp
```

The following command unloads the program module `stcp.cp.pm` in the directory `%eng#m1>system>kernel_loadable_library`.

```
unload_kernel_program
%eng#m1>system>kernel_loadable_library>stcp.cp.pm
```

Related Information

For information about listing and loading kernel programs, see the descriptions of the `list_kernel_programs` and `load_kernel_program` commands. For information on the `bind` command, see the *OpenVOS Commands Reference Manual* (R098).

update_codes

Purpose

The `update_codes` command puts into effect changes made to the files `error_codes.tin` and `user_codes.tin`. Issue this command after you modify `user_codes.tin` or when `error_codes.tin` is modified due to the installation of a new system release.

Display Form

```
-------------------------------- update_codes -------------------------------
No arguments required.  Press ENTER to continue.
```

Command-Line Form

```
update_codes
```

Explanation

The command concatenates `error_codes.tin` and `user_codes.tin` (if this file exists), creates an updated `error_codes.table` file, and creates an updated `error_codes.text` file.

To successfully execute this command, you must have modify access to the directory `(master_disk)>system>error` and default write access to the files in that directory. The `update_codes` command should be issued only from the `(master_disk)>system>error` directory.

If performed during either a major or update installation, you should invoke this command **after** the preboot phase of the installation and **before** you shut down and reboot the module(s).

Related Information

See Chapter 9, ‘‘Customizing Status Codes and Messages.’’

update_default_cmd_limits *Privileged*

Purpose

The `update_default_cmd_limits` command changes the initial and maximum default resource limits for either an existing command process or set of command processes started on a specified module.

Display Form

```
--------------------------- update_default_cmd_limits ------------------------
module:                    current_module
-initial_total_limit:
-initial_heap_limit:
-initial_stack_limit:
-initial_cpu_limit:
-initial_file_limit:
-initial_keep_limit:
-initial_port_limit:
-maximum_total_limit:
-maximum_heap_limit:
-maximum_stack_limit:
-maximum_cpu_limit:
-maximum_file_limit:
-maximum_keep_limit:
-maximum_port_limit:
```

Command-Line Form

```
update_default_cmd_limits module
          [-initial_total_limit number]
          [-initial_heap_limit number]
          [-initial_stack_limit number]
          [-initial_cpu_limit number]
          [-initial_file_limit number]
          [-initial_keep_limit number]
          [-initial_port_limit number]
          [-maximum_total_limit number]
          [-maximum_heap_limit number]
          [-maximum_stack_limit number]
          [-maximum_cpu_limit number]
          [-maximum_file_limit number]
          [-maximum_keep_limit number]
          [-maximum_port_limit number]
```

Arguments

- `module`
  The name or star name of a module for which you want to set the default command limits. By default, the command uses the current module.

- `-initial_total_limit` *number*
  Specifies, in bytes, the initial value for a command's current total virtual-space limit. When this limit is reached, the user heap cannot be increased, the stack space cannot be increased, and no additional shared memory can be used. The default value is `infinity`, which indicates that the command imposes no limit.

- `-initial_heap_limit` *number*
  Specifies, in bytes, the initial value for a command's heap-space limit, which is the number of bytes of process address space that can be devoted to the user heap. When this limit is reached, the user heap cannot be increased. The default value is `infinity`, which indicates that the command imposes no limit.

- `-initial_stack_limit` *number*
  Specifies, in bytes, the initial allocation for a command's stack space, which is the number of bytes of process address space that is devoted to the main stack. The value must be an integer, though infinity is not an acceptable value. The default value is 8,388,608 bytes.

- `-initial_cpu_limit` *number*
  Specifies, in seconds, the initial value for the number of CPU seconds that a command can consume before it is sent a stop-process signal. The default value is `infinity`, which indicates that the command imposes no limit.

- ▶ `-initial_file_limit` *`number`*
  Specifies, in bytes, the initial size of a stream file opened for append or output access. When the limit is exceeded, attempts to write to the file return the error code `e$max_file_exceeded`. The default value is `infinity`, which indicates that the command imposes no limit.

- ▶ `-initial_keep_limit` *`number`*
  Specifies, as a non-negative integer, the initial value for a command's keep limit. The value `0` specifies that the system should not attempt to create a keep module. A non-zero value specifies that the system should attempt to create a keep module, when necessary. The default value is `infinity`, which indicates that the command imposes no limit.

- ▶ `-initial_port_limit` *`number`*
  Specifies the initial value for the port limit, which is the number of ports that can initially be attached by the process. Specify a value, `5` to `4096`. The default value is `4096` (infinity is not an acceptable value).

- ▶ `-maximum_total_limit` *`number`*
  Specifies, in bytes, the maximum value for the address space limit, which is the maximum number of bytes of process address space that may be used by the process. When this limit is reached, the user heap cannot be increased, the stack space cannot be increased, and no additional shared memory can be used. The default value is `infinity`, which indicates that the command imposes no limit.

- ▶ `-maximum_heap_limit` *`number`*
  Specifies, in bytes, the maximum value for the heap space limit, which is the maximum number of bytes of process address space that can be devoted to the user heap. When this limit is reached, the user heap cannot be increased. The default value is `infinity`, which indicates that the command imposes no limit.

- ▶ `-maximum_stack_limit` *`number`*
  Specifies, in bytes, the maximum value for the stack space limit, which is the maximum number of bytes of process address space that is devoted to the main stack. The default value is 132,513,792 bytes. The value can be infinity, but is not recommended.

- ▶ `-maximum_cpu_limit` *`number`*
  Specifies, in seconds, the maximum value for the CPU limit, which is the maximum number of CPU seconds that a process can consume before it is sent a stop-process signal. The default value is `infinity`, which indicates that the command imposes no limit.

- ▶ `-maximum_file_limit` *`number`*
  Specifies, in bytes, the maximum value for the file limit, which is the maximum size of a stream file opened for append or output access. When the limit is exceeded,

attempts to write to the file return the error code `e$max_file_exceeded`. The default value is `infinity`, which indicates that the command imposes no limit.

▶ `-maximum_keep_limit` *`number`*

Specifies, in bytes, the upper limit on default keep module size for commands run in the selected process(es). The value `0` specifies that the system should not attempt to create a keep module. A non-zero value specifies that the system should attempt to create a keep module, when necessary. The default value is `infinity`, which indicates that the command imposes no limit.

▶ `-maximum_port_limit` *`number`*

Specifies the maximum value for the port limit, which is the maximum number of ports that can be attached by the process. Specify a value, `5` to `4096`. The default value is `4096` (infinity is not an acceptable value).

Explanation

The `update_default_cmd_limits` command is a privileged command that sets the default initial and maximum limits for commands starting on a specified module. The command limits control the heap limit, stack limit, total space limit, CPU time limit, stream file size limit, keep module size limit, and the number of attached ports limit. When an administrator changes the module's default command limits, the changes only apply to subsequently-created processes.

The purpose of *command limits* is to control the amount of a resource that an executing program (that is, a command, a process, or command process) can use. A system administrator can adjust the limits to prevent runaway programs from using excessive resources or simply to enforce a degree of fairness on the use of system resources.

The module's *default command limits* are the initial and maximum limits that apply to all processes that are created on the module. These limits, once inherited by each newly-created process, become the process command limits for each process. A system administrator can change these limits with the following commands:

- The `update_default_cmd_limits` command changes the module's default command limits. Any changes apply only to subsequently-created processes.
- The `update_process_cmd_limits` command changes the process command limits.

Each time that OpenVOS executes a program, it copies the process command limits into the program command limits. The program command limits apply only to the currently-executing program.

Each default or process command limit has an initial value and a maximum value, and each program command limit has a current value and a maximum value. When each program command limit is established, the current program value is taken from the initial process value, and the maximum program value is taken from the maximum process value.

The *maximum limits* define the highest size that any command process can use for any of these resources.

The *initial limits* define the limits on a command when it first starts. These limits initialize each command's current limits. These current limits are checked whenever a command wants to change any of its limits. The command can adjust the current limits up or down as it runs, by using the `s$set_current_cmd_limit` subroutine, which is described in the OpenVOS Subroutines manuals. The current limits **cannot** be set greater than their respective maximum limits.

Each time that OpenVOS executes a program, it copies the process command limits into the program command limits. The program command limits only apply to the currently-executing program.

The current value specifies the limit on the amount of a resource that can be obtained by the executing program. A program can dynamically raise its current limit up to its maximum limit by using the `s$set_process_cmd_limits` subroutine. A program can also lower its maximum limit down to its current limit. Only a privileged user, or the root user, can raise one of its maximum limits.

Your system administrator cannot change the **minimum** allowed values for any of the `-maximum_*` arguments. However, your system administrator can change the **maximum** default value for a module for any or all of these arguments. If you are a privileged user, you can increase the value and even specify a value larger than the default maximum value for a module. If you do not specify a value, OpenVOS uses the default maximum limit.

If you issue the `update_default_cmd_limits` command on a module that is running OpenVOS Release 17.2 or later to set default command limits on a module that is running OpenVOS Release 17.1.x or earlier, you receive the message `e$out_of_range` (`1038`) if you attempt to set a value that is larger than OpenVOS 17.1.*x* (or earlier) allows.

Related Information

To list, for a specified module, the default initial and maximum heap limits, stack limits, total space limits, CPU time limits, stream file size limits, keep module size limits, and limits on number of attached ports, use the `list_default_cmd_limits` command.

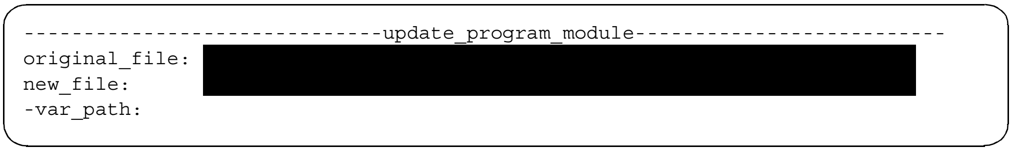
For information about setting the heap, stack, and total (`max_program_size`) size in a bound program module, see the `bind` command description in the *OpenVOS Commands Reference Manual* (R098).

update_program_module

Purpose

The `update_program_module` command reads the records in the variation file, created by the `create_pm_var_file` command, and creates the revised program module.

Display Form



```
------------------------------update_program_module--------------------------
original_file:
new_file:
-var_path:
```

Command-Line Form

```
update_program_module original_file new_file
            [-var_path path_name]
```

Arguments

- ▶ *original_file* **Required**

  The path name of the base program module. Do not specify an extended name.

- ▶ *new_file* **Required**

  The path name of the updated program module. Do not specify an extended name.

- ▶ `-var_path` *path_name*

  Specifies the path name of the variation file. By default, the command uses the name *original_file.var*. Do not specify an extended name.

Explanation

You use the `update_program_module` command with the `create_pm_var_file` command to produce revisions of existing software program modules. The `update_program_module` command reads records in the variation file (created by the `create_pm_var_file` command) and creates the revised program module.

Before the command processes the difference records, it reads the variation file's header record to confirm it is updating the correct file. The program module created by the `update_program_module` command is saved in the same directory in which the command is run. You can use the `update_program_module` and `create_pm_var_file` commands with programs generated in VOS Release 7.0 or later.

Examples

In the following example, the `update_program_module` command creates the `newdump_database.pm` file by updating the `dump_database.pm` file with a previously created variation file.

```
update_program_module
%s#m1>installed>commands>pm>dump_database.pm

%s#m1>installed>commands>pm>newdump_database.pm

Loading the original program module...

Original File: %s#m1>installed>commands>pm>dump_database.pm
         Size: 8 blocks

Clock Time: 1     CPU Time: 0.485291     Page Faults: 31
```

Related Information

See Chapter 7 and the description of the `create_pm_var_file` command earlier in this chapter.

`update_rsnip_site` *Privileged*

Purpose

The `update_rsnip_site` command assigns a site ID and other information to an RSN Internet console server (that is, a console server that provides an RSN-over-IP connection) as well as to the RSN bridge module to which the console server is attached.

Also use this command to configure either an AA-E97900 RSN console server or an AA-E99100 RSN console server that is connected to a modem. (For other console server models, if your connection is through a modem, use the `config_console_server` command.)

NOTICE

You must contact the CAC before using this command.

Display Form

```
------------------------------- update_rsnip_site ---------------------------
site_id:
rsn_connect_password:
RSN_bridge_IP:
private_ip:
-modem:                  no
-sstpc:                  no
-public_ip:
-netmask:
-broadcast:
-default_gateway:
-ntp_server:
-rsn_in_port:
-proxy_addr:
-proxy_port:
-proxy_user:
-proxy_pwd:
-su_pwd:
-ssh_uid:
```

Command-Line Form

```
update_rsnip_site site_id
         rsn_connect_password
         RSN_bridge_IP
         private_ip
         [-modem]
         [-sstpc]
         [-public_ip string]
         [-netmask string]
         [-broadcast string]
         [-default_gateway string]
         [-ntp_server string]
         [-rsn_in_port number]]
         [-proxy_addr string]
         [-proxy_port number]
         [-proxy_user string]
         [-proxy_pwd string]
         [-su_pwd string]
         [-ssh_uid string]
```

Arguments

▶ *site_id* **Required**

The site ID of the RSN bridge module. This ID is also referred to as the RSN site ID. Stratus typically assigns this value, and it is not necessarily identical to the module name or system name. This value must be identical to the value specified by the *site_id* argument of the `start_rsn` command.

NOTE

Before you issue the `update_rsnip_site` command, you **must** coordinate any changes in the value of the *site_id* argument and the value of the *site_id* argument of the `start_rsn` command with the CAC or HUB because the *site_id* values of these two commands and the *site_id* value stored at the HUB must all be identical.

Specify a value of up to 16 characters. The value cannot include the characters `%`, `<`, `>`, `!`, or `#`. The value that you specify must be identical to the value configured at the CAC or support HUB. See "Modifying Your Site ID" on page 10-16 for information on changing your site ID.

▶ *rsn_connect_password* **Required**

The password of the RSN bridge module. Specify a value of up to eight characters.

The value of *rsn_connect_password* is also the value of the RSN CONNECT PASSWORD field in the SITE PARAMETERS screen of the maint_request command's update request. For more information on this argument, see "rsn_connect_password and RSN CONNECT PASSWORD" on page 14-191.

▶ *RSN_bridge_IP* **Required**

The IP address of the RSN bridge module to which the console server is connected. Specify a 4-byte IP address in standard dot notation; do not specify a name. The default value is 10.10.1.1, where 10.10.1 is the default private network (sometimes referred to LAN2 in descriptions of an RSN configuration).

▶ *private_ip* **Required**

The private IP address of the RSN Internet console server. The RSN bridge module, which is on the same private network, uses this address to communicate with the console server. Specify a 4-byte IP address in standard dot notation; do not specify a name. The default value is 10.10.1.200.

▶ -modem CYCLE

By default (no), the module is set to connect to the Internet. Changing the value to yes configures the module to connect to a modem.

▶ -sstpc CYCLE

Use this argument only when directed by the Stratus Customer Assistance Center (CAC) or your authorized Stratus service representative.

▶ -public_ip *string*

The static IP address of the public network, if configured and used by the RSN Internet console server. Specify a name or a 4-byte IP address in standard dot notation. See the Explanation for more information.

If the console server requires a static IP address, you must specify values for this argument as well as the -netmask, -broadcast, and -default_gateway arguments.

▶ -netmask *string*

The subnet mask of the static IP address of the public network, if configured and used by the RSN Internet console server. If the console server requires a static IP address, you must specify values for this argument as well as the -public_ip broadcast, and -default_gateway arguments.

▶ -broadcast *string*

The broadcast address for the subnet specified by the -netmask argument. If the console server requires a static IP address, you must specify values for this argument as well as the `-public_ip`, `-netmask`, and `-default_gateway` arguments.

- ▶ `-default_gateway` *`string`*
  The IP address of the default gateway, if configured and used by the RSN Internet console server.

- ▶ `-ntp_server` *`string`*
  The IP address or host name of the Network Time Protocol (NTP) server. You must specify a value for this argument if the current system is not an RSN bridge module because time on the Internet console server must synchronize with real time; otherwise, the SSL certificate verification process will fail. See the Explanation for more information.

- ▶ `-rsn_in_port` *`number`*
  The port number for the incoming RSN connection. This value is also specified in STCP's `services` file (in the `(master_disk)>system>stcp` directory). The default value is `85`. If you cannot use port 85 (because, for example, an existing application uses it), you must contact the CAC to use a different port number.

- ▶ `-proxy_addr` *`string`*
  The IP address of the proxy server. You can leave this value blank; see the Explanation for more information.

- ▶ `-proxy_port` *`number`*
  The port number of the proxy server.

- ▶ `-proxy_user` *`string`*
  The user ID of the proxy server.

- ▶ `-proxy_pwd` *`string`*
  The password of the proxy-server user.

- ▶ `-su_pwd` *`string`*
  The password of the root user of the RSN Internet console server. You must provide a value for this argument if you have changed the password of the root user of the RSN Internet console server.

- ▶ `-ssh_uid` *`string`*
  The user login name for the SSH tunnel. You must provide a value for this argument if your configuration uses OpenSSH for the RSN connection from the bridge module to the Internet console server (for more information on using OpenSSH, see "Firewall and SSL Concerns For the Internet Console Server" on page 6-11.

Explanation

The `update_rsnip_site` command assigns information to an RSN Internet console server to provide an RSN-over-IP connection. It also configures either an AA-E97900 RSN console server or an AA-E99100 RSN console server that is connected to either the internet or to a modem. The `update_rsnip_site` command, which is located in the `(master_disk)>system>command_library` directory, updates the console-server startup parameters. (If your connection is through a modem, and you are **not** configuring either an AA-E97900 RSN console server or an AA-E99100 RSN console server, use the `config_console_server` command.)

You issue this command in the following situations:

- Before configuring the RSN bridge server for the first time, while installing the module.
- When you have installed a new console server (for example, when replacing a faulty console server).

You must specify values of the *`RSN_bridge_IP`* and *`private_ip`* arguments as 4-byte IP addresses in standard dot notation. You must specify values of the `-ntp_server` and `-proxy_addr` arguments as 4-byte IP addresses in standard dot notation or as names, or leave the values blank. Names must be resolvable by the name servers listed in the `resolv.conf` file. Typically, the `resolv.conf` file is updated automatically when the public address is assigned by the DHCP system (`-public_ip` is left blank). If you specify a value for the `-public_ip` argument, it must be a 4-byte IP address in standard dot notation, and the `resolv.conf` file must be manually edited. To modify the `resolv.conf` file, contact the CAC. If you specify a value for the `-public_ip` argument, you must also specify values for the `-netmask`, `-broadcast`, and `-default_gateway` arguments.

If the RSN Internet console server users a name server, you must configure the following values:

- Specify an NTP server name or IP address using the `-ntp_server` argument of the `update_rsnip_site` command. The command updates a configuration file on the console server with the value.
- Specify values for the name server in the `/etc/resolv.conf` file on the console server; to modify the `resolv.conf` file, contact the CAC.

`rsn_connect_password` and `RSN CONNECT PASSWORD`

The value of the *`rsn_connect_password`* argument is also the value of the `RSN CONNECT PASSWORD` field in the `SITE PARAMETERS` screen of the `maint_request` command's `update` request (see "`SITE PARAMETERS` Screen of the `update` Request" on page 14-66), though the value of the `RSN CONNECT PASSWORD` field is never displayed on the screen. The HUB provides this password when it connects to this RSN bridge module, and the site provides this password when it connects to the HUB. In addition, note the following:

NOTES

1. You must coordinate any changes in the value of the *`rsn_connect_password`* argument and, thus, the value of the `RSN CONNECT PASSWORD` field with the CAC or HUB because these two values must be identical.
2. The value of the *`rsn_connect_password`* argument (and, thus, of the `RSN CONNECT PASSWORD` field) **is different** from the password of the RSN login name that HUB personnel use for interactive login sessions to the site. The RSN login name and password are stored in the `registration_admin` database at the site and at the HUB, so you must also coordinate this information with the HUB.

- When you issue the `update_rsnip_site` command **before** you start the MaintBridgeServer process on an RSN bridge module, the command automatically updates the value of `RSN CONNECT PASSWORD` on the `SITE PARAMETERS` screen to the value specified by the *`rsn_connect_password`* argument.
- If you issue the `update_rsnip_site` command **after** you start the MaintBridgeServer process on an RSN bridge module (for example, if you replace a faulty console server), the *`rsn_connect_password`* that you specify must be identical to the value of `RSN CONNECT PASSWORD` on the `SITE PARAMETERS` screen of the `maint_request` command's `update` request (see "`SITE PARAMETERS` Screen of the `update` Request" on page 14-66).

If the value of `RSN CONNECT PASSWORD` on the `SITE PARAMETERS` screen and the value of the *`rsn_connect_password`* argument are not identical, or if the `PASSWORD ENABLED` field is set to `no`, the console server cannot connect to the HUB.

Related Information

For more information on the RSN, see Chapter 6. See also the following manuals:

- the *OpenVOS STREAMS TCP/IP Administrator's Guide* (R419) for information on STCP's `services` file
- the site planning guide as well as the operation and maintenance guide for your system (as listed in Related Manuals in the Preface) for information about physically connecting the console server to the RSN bridge module

update_software_purchased

Purpose

You use the `update_software_purchased` command to update the software purchased bit information on the specified module. Changing software purchased bits on a live module can be risky, so the Remote Service Network (RSN) stores information about changes sent from the support center, and this command enables the module's software purchased bits information to be updated at the customer's convenience.

Display Form

```
------------------------- update_software_purchased -------------------------
-module: current_module
-delete: no
-ask:    yes
```

Command-Line Form

```
update_software_purchased
            [-module module_name]
            [-delete]
            [-no_ask]
```

Arguments

▶ `-module` *`module_name`*

Specifies the name of the module whose software purchased bit information should be updated. You may also specify a star name. The default value is the name of the current module.

▶ `-delete` CYCLE

Specifies a switch indicating whether to delete the existing software purchased information. The default value is `no`.

▶ `-no_ask` CYCLE

Specifies a switch indicating whether to ask for verification before updating the bits on each module. The default value is `yes`.

Explanation

The `update_software_purchased` command updates, for each specified module, the purchased software changes that are pending.

Examples

The following command updates the purchased software changes on all modules of the current system.

```
update_software_purchased -module *
```

Related Information

See the command description of `display_software_changes` in this chapter.

validate_hub *Privileged*

Purpose

The `validate_hub` command displays a three-digit code unique to a particular site on a particular date. It also displays the site's abbreviated name and the current date.

Display Form

```
-------------------------------- validate_hub -------------------------------
date: current_date
```

Command-Line Form

```
validate_hub [date]
```

Arguments

- `date`
  The date and time for which you want the verification code. The date/time format is *`YY-MM-DD_HH:MM:SS_zone`*. The default is the current date.

Explanation

Use the `validate_hub` command to determine the unique three-digit code for your site on a particular date. To verify that an incoming phone call to your site originates from the HUB, you can request that the caller supply the code for your site.

Examples

The following example illustrates the output generated by the `validate_hub` command.

```
validate_hub
Site_id is smith_co
Validation code on 21-09-03 is 539
```

vtm_config *Privileged*

Purpose

The `vtm_config` command manages and displays Virtual Technician Module (VTM) settings.

> NOTE
>
> The `vtm_config` command is available only on ftServer V 2608, V 4612, V 6616, V 6624, V 6728, and V 2810, V 4820, and V 6832 systems. If you attempt to run this command on an ftServer system that does not support it, the command returns an error message and exits.

Display Form

```
------------------------------ vtm_config ------------------------------
path_name:      path_name
-request_line:
-quit:          no
```

Command-Line Form

```
vtm_config [path_name]
           [-request_line string]
           [-quit]
```

Arguments

▶ *path_name*

The path name for the BMC data file. By default, *path_name* is `(master_disk)>system>bmc_file`.

▶ `-request_line`

Specifies a `vtm_config` request or OpenVOS internal command. If the string contains spaces, apostrophes, or parentheses, enclose it in apostrophes.

If you do not specify a request with the `-request_line` argument when you issue the `vtm_config` command, the command displays the `vtm_config:` prompt.

▶ `-quit` CYCLE

Exits the `vtm_config` command. Use this argument with `-request_line` to return to command level immediately after performing the specified request. The default value is `no`.

Explanation

The VTMs in your system enable authorized system administrators to remotely control, monitor, and diagnose the system through a Web-based interface called the *VTM console*. The VTMs also, optionally, provide a connection that enables the CAC or your authorized service representative to access the system for troubleshooting purposes.

Use the `vtm_config` command to configure or view VTM settings. These VTM settings, which are stored in the BMC data file, are preserved across an OpenVOS upgrade.

You must have write access to the BMC data file in order to update the file. You must be a privileged user in order to query or update the BMC.

The `vtm_config` requests are described in the following sections:

- "The `help` Request"
- "The `network` Request"
- "The `quit` Request"
- "The `service` Request"
- "The `sync` Request"
- "The `user` Request"

The `help` Request

The `help` request displays help information for the `vtm_config` command.

Display Form

```
----------------------------------- help ----------------------------------
-match: █
```

▶ `-match`

Specifies a search term to match.

The `network` Request

The `network` request configures the VTM network settings. The default values are taken from the current VTM configuration.

If you specify the `network` request without any arguments, it displays the current value for each setting.

Display Form

```
---------------------------------- network ---------------------------------
-vtm0host:
-vtm1host:
-dhcp:      yes
-netmask:
-gateway:
-vtm0ip:
-vtm1ip:
-domain:
-vtm0dns:
-vtm1dns:
```

▶ `-vtm0host` *`hostname0`*
Specifies a host name for VTM 0 (that is, the VTM port in I/O element 10).

▶ `-vtm1host` *`hostname1`*
Specifies a host name for VTM 1 (that is, the VTM port in I/O element 11).

▶ `-no_dhcp` CYCLE
Enables or disables DHCP. If this argument is enabled, the `network` request ignores all other arguments except `-vtm0host` and `-vtm1host`.

▶ `-netmask` *`mask`*
Specifies the subnet mask for the network to which the VTM port is connected.

▶ `-gateway` *`gateway`*
Specifies the IP address of the default gateway for the network.

▶ `-vtm0ip` *`stat_addr0`*
Specifies the static IP address for VTM 0's port.

▶ `-vtm1ip` *`stat_addr1`*
Specifies the static IP address for VTM 1's port.

▶ `-domain` *`name`*
Specifies the domain name for the network.

- `-vtm0dns` *`IP_address1`*
  Specifies the IP address of a DNS server.

- `-vtm1dns` *`IP_address2`*
  Specifies the IP address of a second DNS server.

The `quit` Request

The `quit` request exits the `vtm_config` command. This request has no arguments.

The `service` Request

The `service` request configures the access settings for the VTM. The default values are taken from the current VTM configuration.

If you specify the `service` request without any arguments, it displays the current value for each setting.

Display Form

```
-------------------------------- service --------------------------------
-http:        yes
-http_port:
-https:       yes
-https_port:
-ssh:         yes
-ssh_port:
```

- `-no_http` CYCLE
  Enables or disables HTTP.

- `-http_port` *`port_number`*
  Specifies the HTTP port number. You can specify a value of 1 to 65,535 for *`port_number`*.

- `-no_https` CYCLE
  Enables or disables HTTPS.

- `-https_port` *`port_number`*
  Specifies the HTTPS port number. You can specify a value of 1 to 65,535 for *`port_number`*.

- `-no_ssh` CYCLE
  Enables or disables SSH.

- `-ssh_port` *`port_number`*
  Specifies the SSH port number. You can specify a value of 1 to 65,535 for *`port_number`*.

The `sync` Request

The `sync` request synchronizes BMC data. It has no arguments.

The `user` Request

The `user` request changes the VTM console password for an existing user. The default values are taken from the current VTM configuration.

If you specify the `user` request without any arguments, the display does not show the password.

Display Form

```
--------------------------------- user ---------------------------------
-user:       █
-password:
```

- `-user`
  The user whose password will be changed. You must always specify `ADMIN` for this value.

- `-password`
  Changes the specified user’s password.

Examples

In the following example, specifying the `service` request without any arguments returns information about the access settings.

```
vtm_config: service
HTTP enabled         True
HTTP Port            80
HTTPS enabled        True
HTTPS Port           443
SSH enabled          True
SSH Port             22
```

Related Information

See the *Stratus ftServer Virtual Technician Module User’s Guide* (R642) for information about using the VTM console. See *OpenVOS System Administration: Configuring a System* (R287) for information about setting up a VTM.

wait_for_overseer

Purpose

This is a system process command. It is used **only** within `module_start_up.cm` files.

The `wait_for_overseer` command ensures that the Overseer process is running before allowing module startup to proceed.

Display Form

```
------------------------------ wait_for_overseer ---------------------------
module: █
```

Command-Line Form

```
wait_for_overseer [module]
```

Arguments

▶ *`module`*

The module containing the Overseer process. The default value is the current module. You should never specify another value for this argument.

Related Information

See the manual *OpenVOS System Administration: Starting Up and Shutting Down a Module or System* (R282) and the description of the `overseer` command earlier in this chapter. See also the `module_start_up.cm` file shipped with the installation software. This file is stored in the `(master_disk)>system>release_dir` directory.

wait_for_tp_overseer

Purpose

This is a system process command. It is used **only** within `module_start_up.cm` files.

The `wait_for_tp_overseer` command ensures that the module startup waits only the amount of time you specify before allowing the module startup to proceed. If the TPOverseer takes longer to finish log processing, you cannot start TP applications until log processing is completed.

Display Form

```
---------------------------- wait_for_tp_overseer ---------------------------
-time_out: 0
```

Command-Line Form

```
wait_for_tp_overseer -time_out minutes
```

Arguments

▶ `-time_out` *`minutes`*

A number, ranging from `0` to `30`, specifying how many minutes the module startup waits for the TPOverseer to finish log processing before allowing the module startup to continue. When the `-time_out` argument is not specified, the module startup waits until the TPOverseer log processing is complete.

Related Information

See the OpenVOS Transaction Processing Facility Reference manuals and the description of the `tp_overseer` command earlier in this chapter. See also the `module_start_up.cm` file that is shipped with the installation software. This file is stored in the directory `(master_disk)>system>release_dir`.

Appendix A
The OpenVOS Internal Character Code Set

Table A-1 provides the OpenVOS internal character code set. It gives the following information.

- the position of each character within the code sequence
- the hexadecimal 8-bit representation of each character
- the symbol for each character (if it is a printing character)
- the name of each character

Table A-1. OpenVOS Internal Character Sets *(Page 1 of 11)*

Decimal Code	Hex Code	Symbol	Name
0	00	NUL	Null
1	01	SOH	Start of Heading
2	02	STX	Start of Text
3	03	ETX	End of Text
4	04	EOT	End of Transmission
5	05	ENQ	Enquiry
6	06	ACK	Acknowledge
7	07	BEL	Bell
8	08	BS	Backspace
9	09	HT	Horizontal Tabulation
10	0A	LF	Linefeed
11	0B	VT	Vertical Tabulation
12	0C	FF	Form Feed

Table A-1. OpenVOS Internal Character Sets *(Page 2 of 11)*

Decimal Code	Hex Code	Symbol	Name
13	0D	CR	Carriage Return
14	0E	SO	Shift Out
15	0F	SI	Shift In
16	10	DLE	Data Link Escape
17	11	DC1	Device Control 1
18	12	DC2	Device Control 2
19	13	DC3	Device Control 3
20	14	DC4	Device Control 4
21	15	NAK	Negative Acknowledge
22	16	SYN	Synchronous Idle
23	17	ETB	EOT Block
24	18	CAN	Cancel
25	19	EM	End of Medium
26	1A	SUB	Substitute
27	1B	ESC	Escape
28	1C	FS	File Separator
29	1D	GS	Group Separator
30	1E	RS	Record Separator
31	1F	US	Unit Separator
32	20	SP	Space
33	21	!	Exclamation Mark
34	22	“	Quotation Marks
35	23	#	Number Sign
36	24	$	Dollar Sign
37	25	%	Percent Sign
38	26	&	Ampersand

Table A-1. OpenVOS Internal Character Sets *(Page 3 of 11)*

Decimal Code	Hex Code	Symbol	Name
39	27	’	Apostrophe
40	28	(	Opening Parenthesis
41	29	)	Closing Parenthesis
42	2A	*	Asterisk
43	2B	+	Plus Sign
44	2C	,	Comma
45	2D	-	Hyphen, Minus Sign
46	2E	.	Period
47	2F	/	Slant
48	30	0	Zero
49	31	1	One
50	32	2	Two
51	33	3	Three
52	34	4	Four
53	35	5	Five
54	36	6	Six
55	37	7	Seven
56	38	8	Eight
57	39	9	Nine
58	3A	:	Colon
59	3B	;	Semicolon
60	3C	<	Less-Than Sign
61	3D	=	Equals Sign
62	3E	>	Greater-Than Sign
63	3F	?	Question Mark
64	40	@	Commercial “at” Sign

Table A-1. OpenVOS Internal Character Sets *(Page 4 of 11)*

Decimal Code	Hex Code	Symbol	Name
65	41	A	Uppercase A
66	42	B	Uppercase B
67	43	C	Uppercase C
68	44	D	Uppercase D
69	45	E	Uppercase E
70	46	F	Uppercase F
71	47	G	Uppercase G
72	48	H	Uppercase H
73	49	I	Uppercase I
74	4A	J	Uppercase J
75	4B	K	Uppercase K
76	4C	L	Uppercase L
77	4D	M	Uppercase M
78	4E	N	Uppercase N
79	4F	O	Uppercase O
80	50	P	Uppercase P
81	51	Q	Uppercase Q
82	52	R	Uppercase R
83	53	S	Uppercase S
84	54	T	Uppercase T
85	55	U	Uppercase U
86	56	V	Uppercase V
87	57	W	Uppercase W
88	58	X	Uppercase X
89	59	Y	Uppercase Y
90	5A	Z	Uppercase Z

Table A-1. OpenVOS Internal Character Sets *(Page 5 of 11)*

Decimal Code	Hex Code	Symbol	Name
91	5B	[	Opening Bracket
92	5C	\	Reverse Slant
93	5D	]	Closing Bracket
94	5E	^	Circumflex
95	5F	_	Underline
96	60	`	Grave Accent
97	61	a	Lowercase a
98	62	b	Lowercase b
99	63	c	Lowercase c
100	64	d	Lowercase d
101	65	e	Lowercase e
102	66	f	Lowercase f
103	67	g	Lowercase g
104	68	h	Lowercase h
105	69	i	Lowercase i
106	6A	j	Lowercase j
107	6B	k	Lowercase k
108	6C	l	Lowercase l
109	6D	m	Lowercase m
110	6E	n	Lowercase n
111	6F	o	Lowercase o
112	70	p	Lowercase p
113	71	q	Lowercase q
114	72	r	Lowercase r
115	73	s	Lowercase s
116	74	t	Lowercase t

Table A-1. OpenVOS Internal Character Sets *(Page 6 of 11)*

Decimal Code	Hex Code	Symbol	Name
117	75	u	Lowercase u
118	76	v	Lowercase v
119	77	w	Lowercase w
120	78	x	Lowercase x
121	79	y	Lowercase y
122	7A	z	Lowercase z
123	7B	{	Opening Brace
124	7C	|	Vertical Line
125	7D	}	Closing Brace
126	7E	~	Tilde
127	7F	DEL	Delete
128	80	SS1	Single-Shift 1
129	81	SS4	Single-Shift 4
130	82	SS5	Single-Shift 5
131	83	SS6	Single-Shift 6
132	84	SS7	Single-Shift 7
133	85	SS8	Single-Shift 8
134	86	SS9	Single-Shift 9
135	87	SS10	Single-Shift 10
136	88	SS11	Single-Shift 11
137	89	SS12	Single-Shift 12
138	8A	SS13	Single-Shift 13
139	8B	SS14	Single-Shift 14
140	8C	SS15	Single-Shift 15
141	8D		(Not Assigned)
142	8E	SS2	Single-Shift 2

Table A-1. OpenVOS Internal Character Sets *(Page 7 of 11)*

Decimal Code	Hex Code	Symbol	Name
143	8F	SS3	Single-Shift 3
144	90	LSI	Locking-Shift Introducer
145	91	WPI	Word Processing Introducer
146	92	XCI	Extended-Control Introducer
147	93	BDI	Binary-Data Introducer
148	94		(Not Assigned)
149	95		(Not Assigned)
150	96		(Not Assigned)
151	97		(Not Assigned)
152	98		(Not Assigned)
153	99		(Not Assigned)
154	9A		(Not Assigned)
155	9B		(Not Assigned)
156	9C		(Not Assigned)
157	9D		(Not Assigned)
158	9E		(Not Assigned)
159	9F		(Not Assigned)
160	A0	NBSP	No Break Space
161	A1	¡	Inverted Exclamation Mark
162	A2	¢	Cent Sign
163	A3	£	British Pound Sign
164	A4	¤	Currency Sign
165	A5	¥	Yen Sign
166	A6	¦	Broken Bar
167	A7	§	Paragraph Sign
168	A8	¨	Dieresis

Table A-1. OpenVOS Internal Character Sets *(Page 8 of 11)*

Decimal Code	Hex Code	Symbol	Name
169	A9	©	Copyright Sign
170	AA	ª	Feminine Ordinal Indicator
171	AB	«	Left-Angle Quote Mark
172	AC	¬	“Not” Sign
173	AD	-	Soft Hyphen
174	AE	®	Registered Trademark Sign
175	AF	¯	Macron
176	B0	°	Degree Sign, Ring Above
177	B1	±	Plus-Minus Sign
178	B2	^{2}	Superscript 2
179	B3	^{3}	Superscript 3
180	B4	´	Acute Accent
181	B5	µ	Micro Sign
182	B6	¶	Pilcrow Sign
183	B7	·	Middle Dot
184	B8	¸	Cedilla
185	B9	^{1}	Superscript 1
186	BA	º	Masculine Ordinal Indicator
187	BB	»	Right-Angle Quote Mark
188	BC	¼	One-Quarter
189	BD	½	One-Half
190	BE	¾	Three-Quarters
191	BF	¿	Inverted Question Mark
192	C0	À	A with Grave Accent
193	C1	Á	A with Acute Accent
194	C2	Â	A with Circumflex

Table A-1. OpenVOS Internal Character Sets *(Page 9 of 11)*

Decimal Code	Hex Code	Symbol	Name
195	C3	Ã	A with Tilde
196	C4	Ä	A with Dieresis
197	C5	Å	A with Ring Above
198	C6	Æ	Diphthong A with E
199	C7	Ç	C with Cedilla
200	C8	È	E with Grave Accent
201	C9	É	E with Acute Accent
202	CA	Ê	E with Circumflex
203	CB	Ë	E with Dieresis
204	CC	Ì	I with Grave Accent
205	CD	Í	I with Acute Accent
206	CE	Î	I with Circumflex
207	CF	Ï	I with Dieresis
208	D0	Đ	D with Stroke
209	D1	Ñ	N with Tilde
210	D2	Ò	O with Grave Accent
211	D3	Ó	O with Acute Accent
212	D4	Ô	O with Circumflex
213	D5	Õ	O with Tilde
214	D6	Ö	O with Dieresis
215	D7	×	Multiplication Sign
216	D8	Ø	O with Oblique Stroke
217	D9	Ù	U with Grave Accent
218	DA	Ú	U with Acute Accent
219	DB	Û	U with Circumflex
220	DC	Ü	U with Dieresis

Table A-1. OpenVOS Internal Character Sets *(Page 10 of 11)*

Decimal Code	Hex Code	Symbol	Name
221	DD	Ý	Y with Acute Accent
222	DE	Þ	Uppercase Thorn
223	DF	ß	Sharp s
224	E0	à	a with Grave Accent
225	E1	á	a with Acute Accent
226	E2	â	a with Circumflex
227	E3	ã	a with Tilde
228	E4	ä	a with Dieresis
229	E5	å	a with Ring Above
230	E6	æ	Diphthong a with e
231	E7	ç	c with Cedilla
232	E8	è	e with Grave Accent
233	E9	é	e with Acute Accent
234	EA	ê	e with Circumflex
235	EB	ë	e with Dieresis
236	EC	ì	i with Grave Accent
237	ED	í	i with Acute Accent
238	EE	î	i with Circumflex
239	EF	ï	i with Dieresis
240	F0	ð	Lowercase Eth
241	F1	ñ	n with Tilde
242	F2	ò	o with Grave Accent
243	F3	ó	o with Acute Accent
244	F4	ô	o with Circumflex
245	F5	õ	o with Tilde
246	F6	ö	o with Dieresis

Table A-1. OpenVOS Internal Character Sets *(Page 11 of 11)*

Decimal Code	Hex Code	Symbol	Name
247	F7	÷	Division Sign
248	F8	ø	o with Oblique Stroke
249	F9	ù	u with Grave Accent
250	FA	ú	u with Acute Accent
251	FB	û	u with Circumflex
252	FC	ü	u with Dieresis
253	FD	ý	y with Acute Accent
254	FE	þ	Lowercase Thorn
255	FF	ÿ	y with Dieresis

Appendix B

Default IP Addresses for Components of ftServer V Series Systems

Table B-1 lists the default IP addresses for components of ftServer V Series systems.

Table B-1. Default IP Addresses for Components of ftServer V Series Systems

Component	IP Address
Fabric switches	`10.10.1.18` and `10.10.1.19`
Maintenance switch	`10.10.1.75`
RAID arrays in ftScalable Storage systems	`10.10.1.20` through `10.10.1.31`
RSN bridge module	`10.10.1.1`
RSN console server	`10.10.1.200`
StorGate systems	`10.10.1.40`, `10.10.1.41`, `10.10.1.42`, and `10.10.1.43`
Uninterruptible power supply (UPS) units	`10.10.1.160` and `10.10.1.161`
Virtual Technician Modules (VTMs)	`10.10.1.10` and `10.10.1.11`

Index