System Operation

- Console
- Error Log
- System Availability
- Boot Process
- Shutdown
- Servers
- Daemons
- Loading Updates

Error Log

- Written by lots of clients in the environment
  - — Records both normal major environment events and error conditions
- In stable storage
  - — Immediately permanent. Not subject to snapshots.

may get messages about post-snapshot activity that was lost due to crash

Error Log - Format

- Each entry contains 4 fields
  - — Time
  - — Severity
    - --- Normal
    - +++ Warning of normal environment event
    - !!! Error of some sort
    - *** Fatal error
  - — Client - part of system that produces message
  - — Condition name. Identifies specific entry
  - — Comments. Provides additional information

Error Log - Format cont'd...

- Example

```
21:44:18 !!! Job_Manager Bad_Job_Id Id -  224, Count - 1
22:12:29 +++ 87/08/05 Snapshot_Started  1847
22:12:34 +++ Snapshot_Daemon Snapshot_Completed
```

- Date not in each entry

Error Log - Notes

- Fatal messages result in caller stopping in wait service
- Continuation lines are blank up to column 10
- Error Log Daemon
  - — Copies "current" log into a file in !Machine.Error_Logs
  - — Once copied, the log can be read as an ordinary file

Error Log - Notes cont'd...

- Displaying the Current Error Log
  - — From Kernel: Show_Error_Log
  - — From Environment: Op.Internal_System_Diagnosis to EEDB to Kernel
  - — From Environment: Show_Error_Log command
    - Start=0 means to display the end of the error log
- Messages other than --- are displayed on the console
- Throttled to prevent overrun of disk space
-

Error Log - Things to Look For

- Any *** or !!! messages
- Notes about disk, memory, network, or device problems
- Wait service, Out of action ids
- Etc

Other Error Logs

- Session Logs
  - Appear in !Machine.Error_Logs also
  - Have name of user and session, e.g., Phil_S_1
  - Contain lots of messages from the editor about operations
  - When commands fail silently on the user terminal, there are often messages in the session log
  - Very helpful if these are returned with problem reports
  - Logs are reset when user logs in
- Other Error Logs
  - Servers also place output and error files in !Machine.Error_Logs
  - Machine.Initialize creates a log file called !Machine.Error_Logs.Machine_Initialize

System_Availability Subsystem

- Provides report generation from error logs
- Provides programatic interface for reading error logs and producing reports
- Also provides:
  - Machine Id and board information
  - Disk bad-block information
- Future
  - Accounting file reports - will be added soon

System Availability

- Located in !Tools.System_Availability Subsystem
- Reports Generated - System_Report.Generate
  - Availability - Uptime/Downtime by Class
  - Usage - User Load by 1/2 Hour
  - Disk - Used Disk Space each Day
  - Devices - Errors found
  - Daemons - Sizes and Times
  - Outages - Downtimes
  - Trouble - Potential Trouble Areas
  - Advice -

Usage - System usage report

```
Display indicates, for each half hour, number of users
and if disk garbage collection is running (D), and if
system is out of service (X)


        Time           Users  Info
------------------     -----  ----
87/07/15 12:00:00        0  X
87/07/15 12:30:00        0  X
87/07/20 13:00:00        8
87/07/20 13:30:00        8
87/07/20 14:00:00       10
87/07/20 14:30:00       10
87/07/20 15:00:00       11
87/07/20 15:30:00       11
87/07/20 16:00:00       11   D
87/07/20 16:30:00       11   D
87/07/20 17:00:00       10
87/07/20 17:30:00        9
87/07/20 18:00:00        7
87/07/20 18:30:00        6
87/07/20 19:00:00        6
87/07/20 19:30:00        5
87/07/20 20:00:00        5
87/07/20 20:30:00        0  X
```

Daemons

```
Daemon Information Display
--------------------------

Shows at end of day final run of daemon.  This
indicates size before compaction and final size
in pages.


  Date        Ada           File        Action      Directory        DDB
--------  ------------  ------------  ----------  -------------  ------------
87/07/14                              291 -> 291
87/07/15  5369 -> 4974  5153 -> 4761  291 -> 291  7586 -> 7033   1336 -> 1293
87/07/16  5130 -> 4878  4991 -> 4604  291 -> 291  7406 -> 6892   1305 -> 1283
87/07/17  5130 -> 4878  4698 -> 4290  291 -> 291  7874 -> 6419   1289 -> 1210
87/07/18  4608 -> 4509  4359 -> 4241  291 -> 291  6565 -> 6382   1216 -> 1216
87/07/19  4637 -> 4516  4443 -> 4295  291 -> 291  6750 -> 6439   1234 -> 1234
87/07/20  4521 -> 4517  4297 -> 4295   46 -> 46   6449 -> 6439   1235 -> 1235
87/07/21  4700 -> 4455  4544 -> 4316  177 -> 177  7059 -> 6424   1257 -> 1203
87/07/22  4640 -> 3931  4475 -> 3900  177 -> 177  10129 -> 5760  1218 -> 1163
87/07/23  4004 -> 3938  3945 -> 3904  177 -> 177  5827 -> 5768   1171 -> 1171
87/07/24  4121 -> 3997  4094 -> 3989  177 -> 177  6116 -> 5905   1194 -> 1194
87/07/25  4331 -> 4133  4381 -> 4125  177 -> 177  6831 -> 6142   1229 -> 1229
87/07/26  4182 -> 4157  4167 -> 4150  185 -> 185  6227 -> 6176   1234 -> 1234
87/07/27  4298 -> 4163  4279 -> 4197  185 -> 185  6395 -> 6258   1246 -> 1228
87/07/28  4298 -> 4163  4279 -> 4197  185 -> 185  6395 -> 6258   1246 -> 1228
87/07/29  4715 -> 4202  4683 -> 4229  185 -> 185  7389 -> 6292   1292 -> 1266
87/07/30  4398 -> 4259  4451 -> 4284  185 -> 185  6507 -> 6360   1286 -> 1286
87/07/31  4457 -> 4180  4578 -> 3957  185 -> 185  8997 -> 6298   1301 -> 1285
87/08/01  4290 -> 4172  4123 -> 3957  185 -> 185  6532 -> 6268   1290 -> 1290
```

Disk

For each disk, this display indicates the amount of used space after garbage collection has finished that day.

Date	Volume 1	Volume 2	Volume 3	Volume 4
87/07/14		266920		258063
87/07/15	222615	267220	239492	253634
87/07/16	185128	253727	237118	253224
87/07/17	174932	246691	223758	257998
87/07/18	179457	247632	230918	254930
87/07/19	182650	248119	233465	257605
87/07/20	183441	249979	233736	257906
87/07/21	176473	232125	223107	197839
87/07/22	151652	238938	201771	196464
87/07/23	152726	234208	206866	197093
87/07/24	158163	239490	209258	199518
87/07/25	162686	239696	214954	210075
87/07/26	169817	233495	215301	210958
87/07/27	158523	235053	204711	203910
87/07/28	158523	231718	204711	203910
87/07/29	162037	239181	204246	203840
87/07/30	172870	225327	208416	208225
87/07/31	172662	225327	201321	224953
87/08/01	182726	225558	202768	215602

Disk Daemon cont'd...

Disk Daemon Information

Shows, for each day, last disk garbage collector run and amount of used space before and after the collection in megabytes. The time in minutes to do the collection is also shown.

Date	Vol 1	Vol 2	Vol 3	Vol 4
87/07/14		295 -> 267 00:50		301 -> 258 00:29
87/07/15	269 -> 223 00:37	315 -> 267 00:40	284 -> 239 00:37	328 -> 254 00:24
87/07/16	281 -> 185 00:38	295 -> 254 00:51	303 -> 237 00:35	254 -> 253 00:26
87/07/17	235 -> 175 00:32	294 -> 247 00:47	294 -> 224 00:43	284 -> 258 00:29
87/07/18	203 -> 179 00:28	283 -> 248 00:33	284 -> 231 00:29	280 -> 255 00:25
87/07/19	212 -> 183 00:30	249 -> 248 00:29	272 -> 233 00:30	279 -> 258 00:25
87/07/20	187 -> 183 00:26	294 -> 250 00:41	245 -> 234 00:26	259 -> 258 00:24
87/07/21	265 -> 176 00:36	294 -> 232 01:03	281 -> 223 00:31	302 -> 198 00:57
87/07/22	235 -> 152 00:33	269 -> 239 00:34	287 -> 202 00:30	214 -> 196 00:22
87/07/23	165 -> 153 00:26	276 -> 234 00:34	231 -> 207 00:24	203 -> 197 00:19
87/07/24	182 -> 158 00:29	294 -> 239 00:40	227 -> 209 00:26	211 -> 200 00:21
87/07/25	224 -> 163 00:31	248 -> 240 00:30	253 -> 215 00:29	255 -> 210 00:28
87/07/26	184 -> 170 00:29	259 -> 233 00:35	233 -> 215 00:26	227 -> 211 00:22
87/07/27	208 -> 159 00:31	283 -> 235 00:37	254 -> 205 00:28	226 -> 204 00:24
87/07/28	208 -> 159 00:31	325 -> 232 00:43	254 -> 205 00:28	226 -> 204 00:24
87/07/29	266 -> 162 00:41	288 -> 239 00:34	311 -> 204 00:31	274 -> 204 00:31
87/07/30	198 -> 173 00:32	298 -> 225 00:36	255 -> 208 00:28	235 -> 208 00:24
87/07/31	258 -> 173 00:37	298 -> 225 00:36	307 -> 201 00:31	262 -> 225 00:33
87/08/01	211 -> 183 00:34	287 -> 226 00:31	236 -> 203 00:26	247 -> 216 00:28

Devices

```
Device Events


Total Disk messages     -  4
Total Tape messages     -  236
Total Memory messages   -  6
Total Ethernet messages -  982


Log messages concerning disk errors:
-------------------------------------
87/07/16 07:38:31 Disk    ATTEMPT:          ( 258, DATA, 83646, 103)   <--   ( 3,
83834)
    (UNIT => 2, COMMAND => READ)
    RMER1:  ECC_ERROR_IN_READ (DCK) => TRUE
    RMEC1:  ECC_RIGHT_BIT_OF_ERROR => 606
    RMEC2:  ECC_CORRECTING_PATTERN => 00000000001
    RMDC:   CYLINDER => 174
    RMDA:   TRACK => 13
    SECTOR => 5


Log messagés concerning memory errors:
-------------------------------------
87/07/18 10:53:35 Memory    Count of ecc errors since IPL => 1
    Bits with errors (since IPL) =>
    Board 0 (M) Plane 0 Val Bit 16#31#
    ECC events =>
    16#3F04C3200000B8F1#  16#2C89000C57929609#
    Time => 18-JUL-87 10:50:00
    Board 0 (M) Plane 0 Set 16#2# Line 16#4C3#
    Word 16#31# Val Bit 16#31# 1->0 PHYSICAL not TRANSIENT


Log messages concerning Ethernet errors:
-----------------------------------------
87/07/14 13:39:46 Ethernet EXOS CODE 0003 rxmt #1, 2 sec
87/07/14 13:43:10 Ethernet EXOS CODE 0003 rxmt #1, 2 sec
87/07/15 05:15:55 Ethernet EXOS CODE 0115 FF1B0258 <- 3024059
87/07/15 08:01:29 Ethernet TCP/IP Module V3.Sd
87/07/15 08:01:29 Ethernet Internet Address: 89.64.2.3
87/07/15 08:01:29 Ethernet Ethernet Address: 08-00-14-40-02-56
Total messages concerning re-transmits = 776
 726 messages were not displayed.
Too many messages;  additional messages not displayed.
```

Outages

```
System Outages


Each system outage is listed, including the length
of the outage measured from the last successful
snapshot to the elaboration of the environment.
The entered cause or system diagnosed cause and any
comments entered in Shutdown or Explain_Crash are shown.


      Time          Length  Cause              Comments
-----------------   ------  -----   -----------------------------------
87/07/15 07:54:24   00:46   COPS     by SMP.S_1 Cause not entered
87/07/16 12:17:36   00:05   None
87/07/20 22:00:56   01:48   COPS     by SMP.S_1 Cause not entered
87/07/21 07:30:06   00:33   COPS     by OPERATOR.S_1 Cause not entered

87/07/26 16:54:21   01:10   None
87/07/28 21:30:19   01:10   None
87/07/30 11:55:25   01:06   None
```

Availability

```
System Availability Statistics
Total Outages                    -  7
Total Downtime                   - 06:39
Total report time                -  17/17:10
Downtime due to system problems  - 00:00
Downtime due to planned operations - 06:39
Total up time fraction           -   98.4
System availability fraction     -  100.0
```

no cause => scheduled
customers must supply to blame Rational!

Trouble

```
Exception conditions in log - may indicate serious problem
---------------------------------------------------------
87/07/18 03:15:39 !!! Compaction Exception !Lrm.System.Type_Error, from PC...
87/07/20 20:12:52 !!! EEDB Assert_Failure Unexpected Exception: Tasking_Error.
87/07/20 20:12:54 *** Snapshot_Daemon Exception in worker task: ...
*** Calling task (16#DE15C04#) will be stopped in wait service
87/07/20 22:14:31 !!! Mail_Oe Unexpected_Exception Storage_Error (name)
87/07/22 07:37:45 !!! IMAGE_OBJECT UNHANDLED_EXCEPTION  in task 16#3DCCE#
87/07/29 13:48:17 !!! core_editor_task unexpected_death <Exception: ...>
```

Machine Information

- System_Report.Show_Machine_Info

```
I/O Adaptor
    Part Number      -  1
    Serial Number    -  256
    Artwork Rev      -  46
    ECO Level        -  3
    Build Date       - 85/10/11

Sysbus/I/O Controller
    Part Number      -  6
    Serial Number    -  0
    Artwork Rev      -  2
    ECO Level        -  19
    Build Date       - 86/12/02

Sequencer
    Part Number      -  5
    Serial Number    -  0
    Artwork Rev      -  2
    ECO Level        -  5
    Build Date       - 86/12/02
```

System_Availability - bugs

- Bugs
  - Constraint_Error if not files in !Machine.Error_Logs to read
  - Gets into infinite loop if beginning of log is junky. Workaround: Delete first error log file and let it start on second.

Console Interface

- Multiple Clients
  - Each has unique banner
  - ^Z to toggle between clients
  - If a client is waiting for input and is aborted, request still appears. Characters will not be echoed. Type ^Z to get console working.
- Common Clients
  - EEDB
  - Kernel
  - Console command interpreter
  - Tape operator interface
  - Daemons (output only)

Console - Break

- Break key wakes IOP (M200) or cluster manager (M100)
  - Can crash system, redisplay output, or enter debugger
- Be careful about pressing return without knowing what console is asking
  - Type Break-Redisplay first

Boot Process

- Steps on IOP
  - — Load microcode and R1000 registers
  - — Load "wired" memory code segments
  - — Start R1000 CPU running
  - — Send information packet describing system configuration
  - — Start IOP IO kernel running to service R1000

Boot Process cont'd...

- Steps on R1000
  - — Initialize basic machine packages
  - — Initialize low level I/O packages
  - — Initialize and start kernel debugger
  - — Elaborate and start Kernel
  - — Elaborate environment debugger
  - — Elaborate utilities
  - — Start Environment elaborator database (EEDB)

Kernel startup

- Scan disks to check for integrity
- Traverse kernel disk data structures to return system to a consistent state
- Start virtual memory
  - — At this point, page faults can occur

EEDB

- Controls elaboration and sequencing of environment subsystems from configuration
- Elaborates each subsystem in turn
- Once all environment subsystems are elaborated, processes additional commands
  - — Command reference summary available
- Diagnostic configurations can also be run from EEDB

Major Steps in Environment Elaboration

- Object management system abandons uncommitted actions at time of last smapshot
- Object managers perform archive restore if system was shutdown with Archive-on-shutdown
- Editor reads help files
- Deleted code segments are actually destroyed
- Temp heaps are destroyed

Major Steps in Environment Elaboration cont'd...

- Keymaps are read from !Machine.Editor_Data
- !Machine.Devices is created to match the devices that exist on the machine
- Terminal 16 is enabled for login
- Machine.Initialize is started

Machine.Initialize

- Broken into a number of procedures
  - — Initialize_Houskeeping
    - Clears !Machine.Temporary; sets scheduler parameters
  - — Initialize_Daemons
    - Sets daemon parameters
  - — Initialize_Terminals
    - Enables terminals for login - hardwire and telnet
  - — Initialize_Network
    - Boots network controller and starts servers

Machine.Initialize cont'd...

- — Initialize_Servers
  - Starts print spooler and console command interpreter
- — Initialize_Site
  - Reserved for customer use
- — Initialize_Mail
  - Will be used to start mail product
- — Initialize_Cross_Compilers
  - Will be used to set up cross development products
- — Initialize_Print_Spooler
  - Internal machines only - configures spooler

Shutdown

- Op.Shutdown/Schedule_Shutdown
  - Op.Shutdown waits shutdown warning time
  - Controlled with Op.Shutdown_Warning
  - Display with Op.Show_Shutdown_Settings
  - Schedule_Shutdown taskes a time and shuts down then system then
- Shutdown from EEDB
  - Quit command
  - Not as orderly. It warns you
- Break - 0
  - Not orderly. Doesn't take snapshot. No warning to users

Shutdown cont'd...

- Shutdown steps
  - Warn users
  - Disable terminals
  - Force logoff users
  - Kill batch jobs
  - Wait for things to quiesce — from EEDB starts here
  - Abandon actions in progress
  - Delete action manager state
  - Snapshot
  - Crash to DFS

"Unplanned" Crash

- No warning or snapshot
- Several flavors of crash
  - — Software detected crash
    - Kernel assert failure
    - Explicit call to crash microcode
    - Kernel debugger catches exception from kernel
    - Stub kernel debugger (installed in customer machines) then prints warning on the console and crashes the machine
    - Non-stub allows remote debugger to be connected

"Unplanned" Crash cont'd...

- — Microcode crash
  - ucode detects machine or ucode problem and crashes machine
- — Hardware machine check
  - Internal hardware failure detected

cf. explain-crash after crash

Crash Dump

- IOP normally asks you if you want to take the dump
- Other ways
  - — Type ^C to get to CLI> prompt
  - — Type "X Crashdump"
- Mount 2400' tape at 1600 BPI
- Takes about 10-15 minutes

R1000 Configuration information

- Boot/Crash/Maintenance Options
- DFS "cedit" configuration
- EEDB configuration

Boot/Crash/Maintenance Options

- Can be set when key switch in "Interactive" position

- Options

  — Modem Dialout - allows R1000 to initiate dialout

    - Needed for response center to get crash notification

  — Modem Answer - allow R1000 to accept incomming calls modem will always answer but this prevents it from talking

    - Needed to connect remote debuggers

  — IOP auto boot - if false, asks to boot from tape

  — Auto crash recovery - automatically reboots after crash. Not recommended - disk problems

  — Console Break key - can be disabled

  — Are these new defaults - if not, they apply to this boot only!

Boot/Crash/Maintenance Options cont'd...

- Display

  — Crash machine with key switch in "interactive" position

  — Kernel Show_Configuration_Bits command

DFS Configurations

- Specifies options and versions of subsystems up to EEDB
- File name *config
  - Can get list of them with DFS CLI command "dir *config"
- Display and edit with cedit command
  - CLI> x cedit
  - Asks name of configuration to edit and save
  - If you just press returns, no changs will be made
  - General: press return till you see what you want to change, then type the new value
- After fresh DFS load
  - Need to run cedit to change establish hardware configuration
  - Pressing return a bunch of times will do the right

thing

DFS Configurations cont'd...

- Other flags

  — Auto Boot Kernel Debugger

  - If false, elaboration will stop after kernel debugger and environment debugger. This allows debugging of elaboration code. If this gets turned on in a configuration, you cannot boot that configuration without connecting a remote debugger.

  — Wait for Remote Debugging on Crash

  - Only operates with non-stub kernel debugger. Causes debugger to wait for a remote connection if an exception is detected.

  — Call Rational on Crash

  - Causes debugger to call out for software crashes and disk errors

  — Auto Boot Kernel

  - Causes kernel to start virtual memory. If not, kernel elaborates and starts kernel command interpreter, but virtual memory does not start.

  — Auto boot environment elaborator

  - Causes EEDB to elaborate default configuration

- Others are or were used for special debugging and should not be changed.

DFS configurations cont'd...

- Standard
  - Boots to EEDB and elaborates default configuration
- EEDB
  - Boots to EEDB. Doesn't have EEDB elaborate anything
- Kernel
  - Boots to Kernel. Doesn't start virtual memory.

EEDB Configurations

- Specify a list of subsystems to be elaborated
- Configurations are structured in a tree
- There is a default configuration
  - Elaborated when EEDB first starts if auto boot env. elaborator CEDIT configuration bit is set.
  - Display: EEDB: Show_Default
  - Set: EEDB: Default <config name>

EEDB Configuration Operations

- Replacing a subsystem in an existing configuration
  - EEDB: Replace <Configuration> <Subsystem>.X.Y.Z
  - Changes an existing configuration by changing the version of the subsystem.
  - Current configuration can't be elaborated
- Making a new configuration
  - EEDB: Copy <current config> <new config>
  - Then replace subsystems in the new configuration.
  - EEDB: Default <new config>
  - Then shutdown and reboot. New configuration will be ealborated.

EEDB Configuration Operations cont'd...

- Elaborating a configuration
  - EEDB: e <config name> -- e for Elaborate
  - Must be a branch of the currently elaborated tree
- What is running?
  - EEDB: Running
  - Reports all elaborated or partially elaborated configurations

Test Configurations

- Provide various diagnostic tests
  - — Disk_Exerciser
  - — Port_Exerciser
  - — etc_Exerciser

Servers

- Starting
  - — Generally by Machine.Initialize, but can be started anytime
  - — Need to be careful about I/O because starting session may go away
  - — Good idea to put log in !Machine.Error_Logs, but there are access and space issues

Servers cont'd...

- Rational Servers
  - — FTP - for file transfer
  - — Archive Server
    - Processes Archive.Copy requests
    - Server always receives units
    - If the copy source is on the local machine, the command sends the units
    - If the copy source is on the remote machine, the server on that machine sends the units, and the sever on this machine receives them
  - — Console Command Interpreter
    - Processes console login and command execution

Servers cont'd...

- — Print Spooler
  - Local print spooler
  - Runs as a system job; special commands to control it
- — Queue Server
  - Network print spooler

Daemons

- Action Daemon
  - Run frequently (20 min - 2 hours)
  - Does not compact anything so sizes always the same
  - Abandons actions for dead tasks
- Ada, File, DDB, and Directory Daemons
  - Runs daily (nightly) for 1-5 minutes
  - Copies valid data in manager space
  - If size doesn't decrease, exceptions are reported, or the size seems to grow very large, there is a problem

Daemons cont'd...

- Snapshot
  - Runs snapshots; doesn't compact anything
  - Recommended interval is 30 minutes
  - Time for snapshot is proportional to the amount of information modified since the previous snapshot
- Other sources of snapshots
  - Backups
  - Disk garbage collection (per volume)
  - manual command
    - Daemon.Run("Snap")
    - EEDB: sn

Daemons cont'd...

- Archived_Code
  - Manager for code databases of subsystem code views and loaded main programs
- Code_Segment
  - Manages all code segments
  - Size is not interesting
- Configurations
  - Not used
- Error_Log
  - Copies error log to file. Does not compact anything
- User, Group, Session
  - Manages these small objects
  - Fast running and size should be small

Daemons cont'd...

- Link
  - Manages all links on the system
  - Should be small and fast running
- Null_Device, Tape, Terminal
  - These are device managers. Small size and fast running.
- Pipe
  - Manages pipe objects. Small and fast running.
- Image_Tree
  - Starts a job
  - Checks consistency between all Ada unit images and Diana trees
  - Reports errors to error log

Daemon Information

- Daemon.Status
  - — Daemon.Status; -- information on major daemons
  - — Daemon.Status(""); -- information on all daemons
  - — Daemon.Status("Disk");
    - Provides additional information about state of disk garbage collection daemon

Loading Updates

- AK tapes
  - — Read using DFS CLI command: CLI> load
  - — Includes code segment files and DFS configuration files
  - — Automatically sets Standard, EEDB, and Kernel configurations
  - — If alternate configurations were created, they may not be updated.
  - — If changes were made to the standard configurations, they are overwritten.
- AE tapes
  - — Read using EEDB command: EEDB: read
  - — Loads subsystems (segments and descriptive information) and creates configurations
  - — Configuration has the name of the version of the release, for example, D_9_20_2

```
package System_Report is

-- Report generation from system availability information.
-- Provide a variety of reports.

    type Report_Class is (Availability, -- Uptime/downtime by classes
                          Usage,        -- Per half hour, # users, etc.
                          Disk,         -- Used disk space each day
                          Devices,      -- Disk, Mem, Tape, etc errors
                          Daemons,      -- Daemon state sizes and times
                          Outages,      -- System outages and reasons
                          Trouble,      -- Potential trouble areas
                          Advice,       -- Advice on cleaning things up
                          Everything); -- All of above

    procedure Generate (Report_Type : Report_Class := System_Report.Everything;
                        Start_Time : String := "";
                        End_Time : String := "";
                        Log_Directory : String := "!Machine.Error_Logs");


    -- Run a report of the specified type.  Output goes to current
    -- output.  Start_Time null or illegal means "earliest time for
    -- which there is information".  End_Time null of illegal
    -- means "now".  Log_Directory is directory from which error
    -- log files will be read; this is changed mostly for testing.


    procedure Show_Bad_Blocks;
    procedure Show_Machine_Information;

    -- Produce a specific report about some specific subject.

end System_Report;
```

```
procedure Show_Error_Log (Start : Natural := 0; Count : Natural := 30);
-- Start = 0 => show end of log
```

```
with Simple_Status;
with Time_Utilities;
with Bounded_String;

package System_Information is

-- Interfaces to extract information used to produce System_Report output.

    procedure Generate (Start_Time : String := "";
                        End_Time : String := "";
                        Log_Directory : String := "!Machine.Error_Logs";
                        Log_Time : out Duration;
                        Status : in out Simple_Status.Condition);

    -- Must be called prior to using the following operations.  They, then
    -- can be used to read the reduced data.  Log_Time indicates the
    -- actual duration between the first and last entries used in this
    -- report.

    -- Each iterator has a type and returns certain information.
    -- General paradign for each is:
    --
    --      I : xxx_Iterator;
    --      Info : xxx_Information;
    --
    --      Initialize (I);
    --      while not Done (I) loop
    --          Info := Value (I);
    --          -- Do something with Info
    --          Next (I);
    --      end loop;

    type Usage_Iterator is private;
    type Outage_Iterator is private;
    type Event_Iterator is private;
    type Device_Iterator is private;
    type Daemon_Iterator is private;

    procedure Initialize (I : out Usage_Iterator);
    procedure Initialize (I : out Outage_Iterator);
    procedure Initialize (I : out Event_Iterator);
    procedure Initialize (I : out Device_Iterator);
    procedure Initialize (I : out Daemon_Iterator);

    procedure Next (I : in out Usage_Iterator);
    procedure Next (I : in out Outage_Iterator);
    procedure Next (I : in out Event_Iterator);
    procedure Next (I : in out Device_Iterator);
    procedure Next (I : in out Daemon_Iterator);

    function Done (I : Usage_Iterator) return Boolean;
    function Done (I : Outage_Iterator) return Boolean;
    function Done (I : Event_Iterator) return Boolean;
    function Done (I : Device_Iterator) return Boolean;
    function Done (I : Daemon_Iterator) return Boolean;

    type Pstring is access String;  -- Strings are accessed by dereferencing
    --                                 pointers.

    -- Usage information is available for each half-hour during the
```

```
    -- report period.
    type Usage_Information is
        record
            Time : Time_Utilities.Time;  -- Time of this sample
            Users : Natural;             -- # users logged on
            Disk_Running : Boolean;      -- Disk Daemon running
            Outage : Boolean;            -- System is down
        end record;

    -- Outage information is available for each system service outage.
    type Outage_Information is
        record
            Time : Time_Utilities.Time;  -- time of outage
            Length : Duration;           -- length of outage
            Cause : Pstring;             -- Cause entered
            Explanation : Pstring;       -- Explanation entered
        end record;

    type Event_Class is (User_Operation, Exception_Cond,
                         System_Boot, Other_Event);

    -- Event information is available for each "interesting" event.
    -- The Event_Class gives some idea of the what the event is.
    -- The Info is the log entry for the event and has the standard
    -- format for a log entry.
    type Event_Information is
        record
            Time : Time_Utilities.Time;
            Info : Pstring;
            Event_Kind : Event_Class;
        end record;


    type Device_Class is (Disk, Tape, Ethernet, Memory, Other_Device);

    -- Device information is available for each device error or other
    -- event of interest.  Class indicates for which device it is, and
    -- Info is the log entry for the event.
    type Device_Information is
        record
            Time : Time_Utilities.Time; -- Time of entry
            Info : Pstring;             -- Log entry for device
            Class : Device_Class;       -- Class of device
        end record;

    -- Daemon information is available for each run of a daemon.
    -- The information is as listed below.
    type Daemon_Information is
        record
            Time : Time_Utilities.Time;          -- time of start
            Name : Bounded_String.Variable_String (40);-- Daemon name
            Length : Duration;                   -- length of run
            Pre_Size : Natural;                  -- pages at start
            Post_Size : Natural;                 -- pages of state at end
            Explanation : Pstring;               -- Other info
        end record;


    -- The value functions return the actual information for
    -- each value of the iterator.
    function Value (I : Usage_Iterator) return Usage_Information;
    function Value (I : Outage_Iterator) return Outage_Information;
    function Value (I : Event_Iterator) return Event_Information;
    function Value (I : Device_Iterator) return Device_Information;
    function Value (I : Daemon_Iterator) return Daemon_Information;

private
    type Usage_Iterator is new Integer;
    type Outage_Iterator is new Integer;
    type Event_Iterator is new Integer;
    type Device_Iterator is new Integer;
    type Daemon_Iterator is new Integer;


end System_Information;
```

```
with Time_Utilities;
with Simple_Status;

package Log_Reader is

-- Abstraction for reading the system error log.
--
-- Provides an Iterator that automatically crosses log files and also
-- extends into the current active error log.  Thus, log messages
-- can be read right up to the last one issued by the system.
--
-- The date part of the date/time for an entry may be unknown for the first
-- few entries in the log.  In this case, 1/1/1901 is returned.
--
-- Continuation lines are automatically incorporated into each entry so that
-- each call to Next moves to a complete new entry.  Continuation lines
-- are read as part of the Message field and are preceeded by ASCII.lf
-- characters.


    procedure Load_Logs (From_Directory : String := "!machine.error_logs");
    -- Initialize the module.  Must be called before any other operations.
    -- Builds map of log files in the specified directory.


    type Iterator is private;
    procedure Initialize (I : out Iterator;
                          Status : in out Simple_Status.Condition);

    procedure Next (I : in out Iterator;
                    Status : in out Simple_Status.Condition);


    function Done (I : Iterator) return Boolean;
    function Current_Entry (I : Iterator) return String;
    function Current_Time (I : Iterator) return Time_Utilities.Time;
    function Current_Severity (I : Iterator) return String;
    function Current_Client (I : Iterator) return String;
    function Current_Condition (I : Iterator) return String;
    function Current_Message (I : Iterator) return String;
    function Current_File (I : Iterator) return String;

    function Get_Time (Log_Entry : String) return Time_Utilities.Time;
    -- Note that only the HH:MM:SS part of the time is set by this operation
    function Severity (Log_Entry : String) return String;
    function Client (Log_Entry : String) return String;
    function Condition (Log_Entry : String) return String;
    function Message (Log_Entry : String) return String;

    function Number_Of_Log_Files return Natural;
    -- Returns number of log files that exist.  Defined only after call
    -- to Load_Logs.
private

    type Iterator_Data;
    type Iterator is access Iterator_Data;

end Log_Reader;
```

```
with Calendar;
package Daemon is

    -- There are five types of Daemon tasks controlled by this package, their
    -- characteristics and default scheduling:
    --
    --      Snapshot.   Frequent.  ~1 minute slowdown.  Hourly.
    --
    --      Action.     Frequent, unobtrusive.  Every two hours.
    --
    --      Weekly.     Unobtrusive.  Weekly at 2:30 AM.
    --                  Code_Segment Group Session Tape Terminal User
    --
    --      Daily.      Variable, possibly significant interruption.
    --                  Nightly at 3:00 AM.
    --                  Ada DDB Directory Error_Log File Disk
    --
    --      Disk.       Daily or as needed.   Prolonged slowdown.
    --                  Last portion of the Daily run
    --
    -- If no other action is taken, all clients will be scheduled at a
    -- frequency and time normally appropriate.  These schedules can be
    -- changed to suit specific needs.  Note that Disk is included in the
    -- Daily category and will be run with the other Daily Daemons.
    --
    -- Clients that interfere with normal operations warn all users.
    --
    -- There is a group of clients referred to as Major_Clients that are
    -- expected to be of interest in monitoring the state of the machine:
    --     Snapshot, Action, Disk, Ada, DDB, Directory, and File.

    Major_Clients : constant String := "*";

    procedure Run (Client : String := "Snapshot";
                   Response : String := "<PROFILE>");
    -- Cause the named Client to run the specified operation immediately;
    -- Has no effect on the next scheduled run of Client.

    procedure Schedule (Client : String := ">>CLIENT NAME<<";
                        Interval : Duration;
                        First_Run : Duration := 0.0;
                        Response : String := "<PROFILE>");

    -- Sets the interval at which the Client operation will take place.

    procedure Quiesce (Client : String := ">>CLIENT NAME<<";
                       Additional_Delay : Duration := 86_400.0;
                       Response : String := "<PROFILE>");
    -- Reschedule the Client not to run at the next scheduled time.
    -- Equivalent to Schedule with a new First_Run, but the same Interval.
    -- Defaults to a 1-day delay; use Duration'Last for indefinite delay.

    procedure Status (Client : String := "*");
    -- print a formatted display of current status for given Client
    -- Matches on prefix of Client name, "" is prefix of all clients
    -- Major Clients (*): Actions, Ada, DDB, Directory, Disk, File, Snapshot
    -- The Disk Client provides additional information when run separately.

    procedure Warning_Interval (Interval : Duration := 120.0);
    function Get_Warning_Interval return Duration;
```

```
    -- Warning given before starting Daily clients to allow time to Quiesce.

    function In_Progress (Client : String) return Boolean;
    function Next_Scheduled (Client : String) return Calendar.Time;
    function Last_Run (Client : String) return Calendar.Time;
    function Interval (Client : String) return Duration;
    procedure Get_Size (Client : String;
                        Size : out Long_Integer;
                        Size_After_Last_Run : out Long_Integer;
                        Size_Before_Last_Run : out Long_Integer);
    -- Sizes are set to -1 if invalid

    -- Control of the Disk Daemon
    --
    -- The Disk Daemon runs in response to a number of stimuli:
    --
    --   Daemon.Schedule    Runs at priority 6; intended for machine idle.
    --   Daemon.Run         Runs at priority -1; background collection.
    --   Daemon.Collect     Runs at specified priority
    --   over threshold     Starts at priority 0 with escalation
    --
    -- Messages to all users are issued for each of the three explicitly
    -- called collections.  In addition, a message is sent when a Set_Priority
    -- is called and it causes a change in priority.
    --
    -- A background task monitors over threshold situations and sends messages
    -- of interesting events.  Threshold_Warnings (False) allows an
    -- installation-provided job to tailor policy.
    --
    -- Additional control over Disk operations is available in the
    -- Disk_Daemon tools package.

    subtype Volume is Integer range 0 .. 31;
    subtype Collection_Priority is Integer range -1 .. 6;
    -- -1 is the default and implies very low-level background activity
    --  0 guarantees progress in collection but has some effect on response
    --  6 causes collection to take over the machine

    procedure Collect (Vol : Volume; Priority : Collection_Priority := 0);
    -- If this call initiates a collection, it waits for its completion.

    procedure Set_Priority (Priority : Collection_Priority := -1);
    -- Set the priority of a currently running collection to Priority

    procedure Threshold_Warnings (On : Boolean := True);
    -- Cause messages to be sent when collection thresholds are passed.

    --
    -- Control of snapshot messages
    --

    procedure Snapshot_Warning_Message (Interval : Duration := 120.0);
    procedure Snapshot_Start_Message (On : Boolean := True);
    procedure Snapshot_Finish_Message (On : Boolean := True);
    procedure Show_Snapshot_Settings;
    procedure Get_Snapshot_Settings (Warning : out Duration;
                                     Start_Message : out Boolean;
                                     Finish_Message : out Boolean);

    -------------------------------------------------------------------------
    --
    -- Control of the contents and permanence of the operations error log
    --
    --------------------------------------------------------------------------

    type Condition_Class is (Normal, Warning, Problem, Fatal);
    type Log_Threshold is (Console_Print, Log_To_Disk, Commit_Disk);

    procedure Show_Log_Thresholds;
    procedure Set_Log_Threshold (Kind : Log_Threshold; Level : Condition_Class);
    function Get_Log_Threshold (Kind : Log_Threshold) return Condition_Class;



    -- Options on client compactions.
    --
    -- Consistency checking does additional work to assure that the internal
    -- state of the system is as it seems.  This is normally only run when
    -- there are suspected problems.  Consistency checking slows operations
    -- for which it is meaningful by between one and three orders of magnitude.
    --
    -- Access_List_Compaction is the process of removing non-existent groups
    -- from the access lists of objects.  This condition occurs when groups
    -- are removed from the machine.  Access_List_Compaction is only done
    -- for Ada, Directory and File clients.  All other clients reqested will
    -- be silently ignored.  All three must be compacted for any old group
    -- numbers to be freed.
    --
    -- The default is disabled.  The default is restored after
    -- the next appropriate daemon run has completed.

    procedure Set_Consistency_Checking (Client : String := "";
                                        On : Boolean := True;
                                        Response : String := "<PROFILE>");
    function Get_Consistency_Checking (Client : String := "") return Boolean;

    procedure Set_Access_List_Compaction (Client : String := "";
                                          On : Boolean := True;
                                          Response : String := "<PROFILE>");
    function Get_Access_List_Compaction (Client : String := "") return Boolean;

    pragma Subsystem (Os_Commands);
    pragma Module_Name (4, 3932);

end Daemon;
```