Master Index

◆

INFO: Phone cards	
Applicable to:	*Fix:*
References:	*Keywords:* Phone
Originated from: Phl	*Revised by:*

To call an AT&T operator:

Australia	0014.881.011
Austria	022.903.011
Bahamas	1.800.872.2881
Belgium	11.0010
Brazil	000.8010
Br. Virgin Is.	1.800.872.2881
Cayman Islands	1872
Denmark	0430.0010
Dominica	1.800.872.2881
Finland	9800.100.10
France	19 (await dial tone) 00.11
Gambia	001.199.220.0010
Hong.Kong	008.1111
Hungary	171.499
Italy	172.1011
Jamaica	0.800.872.2881
Japan	0039.111
New Zealand	000.911
Norway	050.12.011
Philippines	105.11
Singapore	800.0011
St. Kitts	1.800.872.2881
St. Maart/Saba	800.1011
Sweden	020.795.611
Switzerland	046.05.0011
The Netherlands	06 (await dial tone) 022.9111
United Kingdom	0800.89.0011
West Germany	0130.0010

With a "carte Pastel", to call a France Telecom operator:

Japan	0039.331
The Netherlands	06 (await dial tone) 022.2033
United Kingdom	0800.89.0033
United States	1.800.5.FRANCE (or 1.800.5.372.623)

Using a "carte Pastel", you may give calls in France, by dialing `10` or `36.50`.

Using a "carte Pastel", you may call various countries from France, by dialing:

```
19 (await dial tone) 33.CC
```

where `CC` is the international number of the country (e.g., `31` for The Netherlands).

♦

INFO: International phone calls		
Applicable to:	*Fix:*	
References:	*Keywords:* Phone	
Originated from: Phl	*Revised by:*	

When giving an international phone call, you will first need to know the prefix for international calls in the country you are in, and the country code of the country you are calling:

Country	International Prefix	Country Code
Austria	00	43
Belgium	00	32
Brazil	00	55
Canada	011	1
Cyprus	00	357
Denmark	009	45
East Germany	00	37
Finland	990	358
France	19 (await dial tone)	33
Great Britain	010	44
Greece	00	30
Ireland	16	353
Italy	00	39
Jugoslavia	99	38
Libya		218
Luxembourg	00	352
Malta		356
Morocco		212
Norway	095	47
Portugal	00	351
Spain	07	34
Sweden	009	46
Switzerland	00	41
The Netherlands	09	31
United States	011	1
West Germany	00	49

♦

INFO: Rational Europe phone numbers and addresses	
Applicable to:	*Fix:*
References:	*Keywords:* Address, Phone
Originated from: Mcf	*Revised by:*

Paris

Rational SARL
18, rue Gounod
F-92210 Saint-Cloud
France
Phone: +33(1)47.71.72.32
Fax : +33(1)47.71.79.77

FAURE, MARIE-CHRISTINE (MCF) - Administrative Assistant
Office address above
Home: +33(1)43.80.26.39

LEROY, PASCAL (PHL) - Technical Consultant
Office address above Aerospatiale
Home: +33(1)45.04.91.54

MUENIER, MICHEL (MCM) - Account Representative
Office address above French customers
Home: +33(1)46.26.39.74

MULLER, PIERRE-ALAIN (PAM) - Technical Consultant
Office address above European Mini Response
Home: +33(1)46.02.76.82 Center

RIPKEN, DR. KNUT (KR) - Managing Director
Office address above Rational Europe
Home: +33(1)47.71.77.17 Marketing Representative
all european customers

KRUCHTEN, DR. PHILIPPE (PBK) - Technical Consultant
4, rue du General Offenstein Bofors, Terma
67100 Strasbourg, France
(O) +33.88.79.30.23
(H) +33.88.79.23.77
(Fax) +33.88.79.21.12

Scandinavia

Rational Scandinavia AB
Box 21153, S:t Eriksgatan 115
S-100 31 Stockholm
Sweden

Phone: +46(833)59.69
Fax: +46(831)94.45

HADEN, STEVEN (SEH) - Technical Consultant
Office address above Bofors, Terma
Phone @ BEAB: +46(758)84.159
Home phone: +46(758)39.941

VESTER, MIKAEL (MV) - Technical Consultant
Office address above Bofors, Terma
Phone @ BEAB: +46(758)84.158
Home Phone: +46(758)70.286

United Kingdom

Rational Techology Ltd.
No office address

LAMBERT, DR. JOHN (JEL) - Technical Consultant
P.O. Box 225 Ferranti, Marconi
Hove
E. Sussex BN3 1D?
United Kingdom

Street Address:
Flat 5
56 Cambridge Road
Hove
E. Sussex BN3 1D?
United Kingdom
Phone: +44(273)20.47.33
Fax: +44(273)25.37.2

FAXes cannot be sent to JEL if he is on-line. If FAX is urgent, please send him a mail message indicating that he needs to log off.

STEED, HOWARD (HTS) - Technical Consultant
31, Gibbs Field Assisting JEL with
Thorley Park Ferranti, Marconi
Bishop's Stortford
Hertfordshire, CM23 4EY
United Kingdom
Phone: +44(279)50.58.41

West Germany
Rational GmbH
no office address

KLEIN, ANNE (ANNE) - Technical Consultant
Gozbert Strasse, 6 MBB, Contraves, Signaal
D-8000 Munchen 90

West Germany
Phone: +49(89)692.42.46

WEHRUM, DR. ROLF-PETER (RPW) - Technical Director
Habenschadenstr. 6/B Rational GmbH
D-8023 Pullach im Isartal
West Germany Technical Consultant
Phone: +49(89)793.88.66 MBB
Fax: +49(89)793.86.01

PRINS, LARS (LHRP) - Technical Consultant
Karwendelstrasse 34 Siemens
8000 Munchen 70
West Germany
Phone: +49(89)725.94.75

◆

INFO: How to dial out from an home-office R1000	
Applicable to:	*Fix:*
References:	*Keywords:* Kermit, R1000
Originated from: Jmk	*Revised by:*

To dial out to an external service via a 2400 bauds modem from an R1000, execute:

```
Kermit.Call ("Dial_24");
Kermit.Options.Physical.Bits := 7;
Kermit.Connect (Log => ">>Log File Name<<");
Def (">>Log File Name<<");
```

To dial a number:

```
D 9-<<the number>>
```

When you want to hang up the phone, type [Ctrl][C] and then [C].

If you anticipate you will want to transfer files or refer to your local R1000 without hanging up the phone, execute:

```
Kermit.Call ("Dial_24");
delay Duration'Last;
```

Note the port number it chooses, and interrupt the resulting job. While this job is running, you may use the following commands:

- To interract with the bulletin board:

```
Kermit.Port := >>Port Number<<;
Kermit.Options.Physical.Bits := 7;
Kermit.Connect (Log => ">>Log File Name<<");
```

- To download a file:

```
Kermit.Port := >>Port Number<<;
Kermit.Options.Physical.Bits := 7;
Kermit.Receive (...)
```

- To upload a file:

```
Kermit.Port := >>Port Number<<;
Kermit.Options.Physical.Bits := 7;
Kermit.Send (...)
```

When your session is complete, `Job.Kill` the background job that executed `Kermit.Call`.
◆

INFO: Development contacts by product

Applicable to: 11-OCT-90	*Fix:*
References:	*Keywords:* Phone
Originated from: Gbd, Phil	*Revised by:*

TOPIC	PEOPLE	PHONE
CPU Hardware	MBD	3824, 3838
IO Hardware	BWB, MBD	3815, 3824, 3838
PDP/11 hardware	MBD	3824, 3838
PDP/11 software	NCE	3789
IOP ethernet	BWB, MBD	3815, 3824, 3838
Rational_Term	CJJ	3823, 3838
Facist_Term	MBD	3824, 3838
Printer	MBD	3824, 3838
Disk	DJL	3828, 3838
Tape	DRL	3809
Comm	BWB, MBD	3815, 3824, 3838
Rxi	GEB	3794
Tbu	PKS	3787
Mc68020_Bare	RFG, WNM	3708, 3783
Mc68020_Os2000	MDW, JST	3786, 3784
Mc68020_Hp_Unix	RFG, MDW, JST	3708, 3786, 3784
Rdf	RJS, SRP, JLS	3836, 3835
Rdf_2167a	RJS, DRS	3836, 3782
Design_Tools	RJS, SRP	3836, 3835
Document_Tools	SRP, RJS	3835, 3836
Mail	JMK, LOREN	3812, 3791
Net	JMK, CLP	3812, 3807
Environment	TRW, PHIL	3745, 3749
R1000_Debugger	PEG, AV	3781, 3760
Code_Generator	RCP, SWB	3755, 3748
Semantics	HJL, TRW	3753, 3745
Core_Editor	TRW, LOREN	3745, 3791
Ada_Editor	TRW, SDJ	3745, 3796
Switch_Editor	TRW, LOREN	3745, 3791
Search_List_Editor	TRW, LOREN	3745, 3791
Help_Editor	TRW, LOREN	3745, 3791
Links_Editor	TRW	3745
Text_Editor	LOREN, TRW	3791, 3745
Jobs_Editor	LOREN, TRW	3791, 3745
Window_Editor	LOREN, TRW	3791, 3745
Window_Io	LOREN, TRW	3791, 3745
Activity_Editor	TRW	3745
Diana_Editor	RK, TRW	3797, 3745
Command_Editor	RK, TRW	3797, 3745
Parser	SDJ, TRW	3796, 3745
Pretty_Printer	SDJ, TRW	3796, 3745
Io	MARLIN, JIM	3805, 3795
Cmvc	TOM, CBH	3804, 3802

Work_Orders	TOM, DRK	3804, 3793
Compose	SRP, RJS	3835, 3836
Utilities	PHIL, TRW	3749, 3745
Speller	DRK	3793
Archive	HJL, TRW	3753, 3745
Access_Control	MARLIN, PHIL	3805, 3749
Directory	TRW, HJL	3745, 3753
Os	GPA, MARLIN	3806, 3805
Support	DEG	3646
Remote_Debug	BLS, WENDY	3788, 3751
Microcode	DJL	3828
Software_Catalog	JLS	3706
Training	MRB, FMJ	3673, 3757
Documentation	MRB, SJL	3673, 3674
On_Line_Help	MRB, SJL	3673, 3674
Prs	PHIL, SMC	3749, 3831
Rdf_2167	RJS, SMC	3836, 3831
Print_Spooler	CLP	3807
RTI	DRS	3782
RPI	SRP, WENDY	3835, 3751

◆

INFO: Driving to Terma	
Applicable to:	*Fix:*
References:	*Keywords:* Terma, Travel
Originated from: Pbk	*Revised by:* Pbk

Address is:

```
Terma Elektronik AS
Markaervej 2
DK-2630 TAASTRUP
+45-42-521513 (voice)
+45-42-525758 (fax)
```

How to get there is more challenging: I found the place difficult to find, especially by car, on your own. Even taxi drivers need a little help to finally locate the place.
With the odometer set to 0 in Hertz parking lot at Kastrup (Copenhagen airport):

```
- km  0 :  take Lufthavnsvej direction Kobenhavn (=Copenhagen)
- km  3 :  at the water tower take road E66v slightly on your right
- km 4.4:  turn left at traffic light, E66 direction Halsskov
- km 7.7:  left E66 direction Halsskov
- km 17 :  take road E4n direction Roskilde
- km 19 :  exit the E4 to take the 21 west towards Roskilde
- km 24.5: north 21
- km 27 :  take direction Hoeje Tastrup
- km 29.5: exit 243 Hoeje Tastrup (Notice the 3 windmills)
- km 30 :  turn left at traffic light
- km 31 :  turn left again at traffic light, leaving windmills
           at left hand, in front of the IKEA store, in street named
           Helgeshoej Allee
- km 31.2: turn right just after IKEA in Markaervej. Drive 100 meters
           turn left again. After 80 meters, you will see on the left
           hand side "Dansk Radio" and a small "Terma" sign at the
           street level. Enter, and drive all the way to the end of
           the yard. Turn right. Park where is says "Gaester". There it is !
```

◆

INFO: Driving to Signaal	
Applicable to:	*Fix:*
References:	*Keywords:* Signaal, Travel
Originated from: Pbk	*Revised by:*

From Schiphol (Amsterdam airport) to Hengelo by car:
With the odometer reset to zero in Hertz parking lot:

- km 0 follow exit signs (exit = Uitgang)
- km 1 take A4, direction Amsterdam (north east)
- km 5 take A9, direction Utrecht; drive through Amstelveen (east)
- km 17 take a2/A9, direction Amsterdam/Amersfoort (north north west)
- km 20 take A9, direction Amersfoort (north east)
- km 26 take finally A1, direction Amersfoort/Hengelo,(east) then go on on A1
- km 158 take A1/A35, south direction Hengelo/Enschede
- km 162 take A35, south, direction Hengelo-Zuid

For Delden and Carelshaven hotel, exit Delden at km 165, take N346 west hotel is on your left just at the entrance of the village.
For Signaal, exit "Twentekanaal" industrial estate, at km 168, east direction hengelo centrum, Signaal is 1 km further on the right hand side; park before the gate, visitor cannot drive in.

From Amsterdam (downtown):

- exit south east, by A1, then follow directions above after km 26.

Drive takes 1 hour 25 minutes when no traffic, 2 hours with traffic (eg. friday afternoon). Speed limit is 120 km/h.

♦

INFO: Completing the Repairable Parts Tag	
Applicable to:	*Fix:*
References:	*Keywords:* Repair Tag, Spare
Originated from: Mbf	*Revised by:*

This card describes how to fill in a Repairable Parts Tag.

Date: The date you use this part

Part#: The part on the piece of equipment used. Also, in some cases, the part number is on box you received the part in. The shipping authorization lists the part number on it. Remember all CPU Boards start with #230-.

Serial#: Not all parts have serial numbers, but every board and even some vendor boards do. Please put this on the tag when it is available.

Customer Support Spare Box: You received the part from the Rational home office.

Internal Spare Box: For RACS only to fill in.

Region Office Spares Box: You received the part from your office or another Region office (like the Eastern Region, Western Region, etc).

On-Site Spare Box: For our international accounts and maybe some other areas in the future.

Other Box: From anywhere else that you may have received a part.

From: The customer or machine name (for internal RACS only) not only the cluster id number. If you would like to include the cluster id you may.

Name: Your name

Status: This area is for the problem/status of the broken part. Please enter a short description of the problem/status/errors, etc. that was the cause for replacement. Please do not only put the SPR# in the area: our repair techs in manufacturing need to know what is wrong with the boards so that they can fix them!

Repair Results: This area is for our internal repair techs to fill in.

Also, if the future it is not necessary to fill out an SAR for replacing a part only. The Repairable Parts Tag is the most important paperwork necessary and must be filled out and returned with the parts to be repaired.
The only time a SAR is necessary is if the parts replacement was billable.
♦

INFO: Memory ECC error info needed on repair tags	
Applicable to:	*Fix:*
References:	*Keywords:* Board, Ecc Error, Memory, Repair Tag
Originated from: Mam	*Revised by:*

When filling out a repair tag on a memory board with ECC errors all you need is the plane and the bit numbers.
You will get it from the error log so do it before you shutdown or after reboot. Typically, with memory ECC errors you will be able to reboot, and can schedule a shutdown.

So from the following ECC error message:

```
04:02:13 *** Memory ECC_error
         Count of ecc errors since IPL => 1
         Bits with errors (since IPL) =>
         Board 1 (L) Plane 1 Val Bit 16#06#
         ECC events =>
         16#3F01847000000446#  16#2CCB0041CB230B3E#
         Time => 23-MAY-88 03:52:22
         Board 1 (L) Plane 1 Set 16#7# Line 16#184#
         Word 16#08# Val Bit 16#06# 1->0 PHYSICAL not TRANSIENT
```

This is the information needed:

```
Plane 1 Val Bit 16#06#
```

♦

INFO: Bibliography about Structured Analysis	
Applicable to:	*Fix:*
References:	*Keywords:* Book
Originated from: Ars	*Revised by:*

- Structured Analysis and System Specification -- Tom DeMarco

  This is the "classic" text on Structured Analysis, it is well written, covers all the important aspects of Structured Analysis, although the examples are MIS-oriented but still clearly convey the structural concepts

- Essential Systems Analysis -- McMenamin and Palmer

  This is the refined version of DeMarco's original text; you can think of it as SA Version II; it is the result of practical experience gained through teaching and consulting on the practical application of the Structured Analysis methodology; it introduces some new analysis components and refines the definition of the previous ones as well as describing a more pragmatic analysis process

- Structured Development for Real-Time Systems -- Mellor and Ward

  Volume 1 Introduction & Tools
  Volume 2 Essential Modeling Techniques
  Volume 3 Implementation Modeling Techniques

  This is the first published text dealing with extending Structured Analysis to model or capture real-time requirements; it is a pretty good treatment but does not warrant three separate volumes, I think this was a ploy to get the beginning stuff published before they finished the conclusion, or triple the income they would have made on a single book

- Strategies for Real-Time System Specification -- Hatley and Pirbhai

  This book came out subsequent to Mellor and Ward; it does not really present anything new, it's just different; if Mellor and Ward are influenced by System Engineering concepts then this approach is influenced by electrical engineering concepts; it is interesting to note that Boeing adopted the initial version of this methodology as a corporate standard for the development of software for the 7J7 program; as a result most CASE tool vendors have implemented the notation.

  The 7J7 program was postponed (indefinitely), Hatley was with Lear Siegler which is now part of Smith Industries, Phirbhai is now a rug salesman in New York...

- Systems Development Without Pain: a users's guide to modeling organizational patterns -- Paul Ward

  Ward is very active consulting on requirements analysis and design methods in the Northeastern portions of the US; this is his latest published stuff.

- Fundamental Concepts of Information Modeling -- Matt Flavin

This is one of the "classics" on Information Modeling, also known as Entity Relationship Modeling. This book elaborates on Peter Chen's original work which was published only as a paper. This takes a slightly different perspective on analysis by initially focusing on data and describing its attributes and relationships and has become the foundation for many OOSA approaches.

- Object-Oriented Analysis -- Peter Coad and Edward Yourdon

  Has had mixed reviews. Probably an attempt by Yourdon to "...get with the program..."

- Object-Oriented Systems Analysis. Modeling the World in Data -- Shlaer and Mellor

  This is a very simplistic "picture book" approach to describing an evolutionary/hybrid Structured Analysis-Information Modeling approach to requirements analysis. Its style tends to put people off, but the methodology has been applied successfully.

◆

INFO: Preventive Maintenance procedures	
Applicable to:	*Fix:*
References:	*Keywords:* Frus, Preventive Maintenance
Originated from: Seh	*Revised by:*

This is the description of the general Preventive Maintenance operations that have to be performed on-site, every 3 months.

First take a snapshot:

```
EEDB: snapshot
```

Once the snapshot has completed, reboot the machine; on the operator console, hit the [Break] key:

```
Please enter
0 => Restart system
1 => Ignore break key
2 => Redisplay recent console output
3 => Enter debugger
Enter option : 0 [CR]

Do you really want to crash the system [N] ? y [CR]
R1000 System Crash detected at PC=000026A4, error code = 08
Operator console break key
Restarting R1000-200 on October 31th, 1989 at 8:09:34

Change BOOT/CRASH/MAINTENANCE options [N] ? y [CR]
Enable MODEM dialout [N] ? [CR]
Enable MODEM answer [Y] ? [CR]
Enable I/O Processor AUTO BOOT [Y] ? [CR]
Enable R1000 AUTO BOOT [Y] ? N [CR]
Enable AUTO CRASH RECOVERY [Y] ? [CR]
Enable CONSOLE BREAK KEY [Y] ? [CR]
Are your answers correct [Y] ? [CR]
Are these new defaults [N] ? [CR]
Booting I/O Processor
IOP Bootstrap Version 2.0
(...)
Restarting system after operations console break key
Do you want to make a crash dump tape [N] ? [CR]
Enter name of configuration to boot [STANDARD] : diag [CR]
Enter CLI [N] ? [CR]
READ_NOVRAM_DATA.TYP (...)
Microcode file [diag] : [CR]
File  : DIAG.M200_UCODE
Bound : July 12, 1986  0:32:20
Delta : (...)
Do you want to enter the CLI prior to starting the machine [N] ? y [CR]

CLI> x expmon
4  8MB MEMORY BOARDS IN PROCESSOR - TOTAL OF 32 MEGABYTES.
```

```
EM> rd
EM> poll_all
NO MACHINE CHECKS DETECTED
```

Wait for approximately 1 minute.

```
EM> poll_all
SEQ HAS DETECTED A MACHINE CHECK
EM> sm
HALT - NEXT UPC IS 0103
NO ECC ERRORS HAVE BEEN DETECTED SINCE THE LAST BOOT
EM>
```

Be sure that the micro PC is 103.

```
EM> bye

CLI>
CLI> x rdiag
READ_NOVRAM_DATA.TYP. (...)
DIAG> test/3 all
```

This takes about 20 minutes.

```
Running FRU P1DCOMM
Running FRU P1IOC
Running FRU P1VAL
Running FRU P1TYP
Running FRU P1SEQ
Running FRU P1FIU
Running FRU P1MEM
Running FRU P1SF
Running FRU P2IOC
Running FRU P2VAL
File  : PHASE2_MULT_TEST.M200_UCODE
Bound : July 16, 1986  14:31:44
Delta :(...)
Running FRU P2TYP
Running FRU P2SEQ
Running FRU P2FIU
Running FRU P2MEM
Running FRU P2UADR
Running FRU P2FP
File  : FPTEST.M200_UCODE
Bound : July 15, 1986  19:02:48
Delta :(...)
Running FRU P2EVNT
Running FRU P2STOP
Running FRU P2ABUS
File  : ABUS_TEST.M200_UCODE
Bound : July 22, 1986  13:27:21
Delta :(...)
Running FRU P2CSA
Running FRU P2MM
Running FRU P2COND
Running FRU P2UCODE
```

```
File   : P2UCODE.M200_UCODE
Bound  : August 6, 1986  16:16:27
Delta  : (...)
Running FRU P3RAMS
Running FRU P3UCODE
File   : P3UCODE.M200_UCODE
Bound  : August 6, 1986  13:50:42
Delta  : (...)
PASSED
DIAG> quit

CLI> force_uload
CLI> boot
```

You may also hit the White Button. If you do so, don't forget to disable the R1000 auto boot.

```
Enter name of configuration to boot [STANDARD] : eedb[CR]
Enter CLI [N] ? n [CR]
File   : 6_72.M200_UCODE
Bound  : October 31, 1988  10:25:19
Delta  : (...)
              Copyright 1984, 1985, 1986, 1987, 1988 by Rational.
(...)

=====>> CONFIGURATOR <<=====
starting diagnosis of configuration
starting virtual memory system
the virtual memory system is up
(...)
```

This takes about 45 minutes, depending on the disk capacity used.

```
=====>> Elaborator Database <<=====

EEDB: elaborate  disk_exerciser
DISK_EXERCISER.9.0.0D          8/19/87 18:21:26

=====>> Disk Exerciser <<=====
For how many minutes to you want this test to run? 5
Test will end at:  9: 57: 52 on  10/ 31/ 1989
=====>> Disk Tester <<=====
(...)
Do you have a hardcopy terminal attached? [n] [CR]
Do you want to specify test options? [n] [CR]
Unit  0 is mapped to volume 1, HDA number F88D_502153, size = ( 745, 27, 66)
Unit  1 is mapped to volume 2, HDA number F88D_501878, size = ( 745, 27, 66)
Unit  2 is mapped to volume 3, HDA number F88D_503014, size = ( 745, 27, 66)
Running test SEEK_PROFILE
Pass #  1, IO # 127     ( 4, DATA, 337882, 1) <== ( 3, 0)
(...)
Pass #  5, IO # 21971     ( 4, DATA, 337882, 1) <== ( 3, 0)
=====>> Disk Exerciser <<=====
Disk Test done
=====>> Elaborator Database <<=====
EEDB:  unelaborate disk_exerciser
Subsystem: [CR]
```

```
Unelaborated DISK_EXERCISER.9.0.0D
```

Run the Tape_Exerciser only if tape unit 0 is *not* an Exabyte. Exerciser an Exabyte can take *days*.

```
EEDB: elaborate tape_exerciser
TAPE_EXERCISER.9.0.0D          8/19/87 18:18:46

=====>> Tape Exerciser <<=====
For how many minutes to you want this test to run? 1
Test will end at:  10: 9: 5 on  10/ 31/ 1989
=====>> Tape Tester <<=====
(...)
Select the test you want by number, or 0 to run them all in order :
First pass is abbreviated, subsequent passes will take longer.
which test? (0..8) :0
Beginning test 1
Test 1 done.
Beginning test 2
Test 2 done.
Beginning test 3
Test 3 done.
Beginning test 4
Test 4 done.
Beginning test 5
Test 5 done.
Beginning test 6
Test 6 done.
Beginning test 7
Test 7 done.
Beginning test 8
Test 8 done.
=== End of Pass 1 for unit number 0
=====>> Tape Exerciser <<=====
Run again? [N] : [CR]
Tape Test done
=====>> Elaborator Database <<=====
EEDB:  unelaborate tape_exerciser
Subsystem: [CR]
Unelaborated TAPE_EXERCISER.9.0.0D
```

Note: If for some reason you need to kill the Tape_Exerciser, on a Fujistsu M244X tape drive, open the door. If you then see a message saying it will retry in 5 seconds, wait 6 seconds before closing the door; you will then get an option to terminate Tape_Exerciser.

Run the Port_Exerciser only if the machine is a Series 100 or 200 *and* you have loopback connectors. Plug the loopback connectors on the ports you want to test. It is recommended that you test one port in each block of 8 ports on the Communication Panel.

```
EEDB: elaborate port_exerciser
PORT_EXERCISER.9.0.0D          10/23/87 17:50:35

=====>> Port Test Instance <<=====
By default, we transmit on ports 18 .. 48
and expect loop back plugs on the first 3 ports
Nonexistent ports are ignored
Do you want to change this? [No] :yes
```

```
Ports in the range 16 .. 48 may be tested. Nonexistent ports are ignored
First port number [ 18] :16
Last port number [ 48] :16
Do you want to also test receive? [No] :yes
First looped back port [ 16] : [CR]
Last loop back port [ 16] : [CR]
Drop RTS at the end of the test (for printer switch) [No] [CR]
For how many minutes to you want this test to run? 1
Test will end at:  10: 18: 30 on  10/ 31/ 1989
=====>> Port Tester <<=====
port 16: Line terminated: 0 total errors out of 38745 bytes
=====>> Port Test Instance <<=====
Port Test done
Do you want to start over? [No] :yes
By default, we transmit on ports 16 .. 16
and expect loop back plugs on the first 3 ports
Nonexistent ports are ignored
Do you want to change this? [No] :yes
Ports in the range 16 .. 48 may be tested. Nonexistent ports are ignored
First port number [ 16] :31
Last port number [ 16] :31
Do you want to also test receive? [No] :yes
First looped back port [ 31] : [CR]
Last loop back port [ 31] : [CR]
Drop RTS at the end of the test (for printer switch) [No] [CR]
For how many minutes to you want this test to run? 1
Test will end at:  10: 21: 19 on  10/ 31/ 1989
=====>> Port Tester <<=====
port 31: Line terminated: 0 total errors out of 38745 bytes
=====>> Port Test Instance <<=====
Port Test done
Do you want to start over? [No] : [CR]

=====>> Elaborator Database <<=====
EEDB:  unelaborate port_exerciser
Subsystem:
Unelaborated PORT_EXERCISER.9.0.0D
EEDB: show_default
Default configuration is D_10_20_0

EEDB: elaborate d_10_20_0
NETWORK.10.0.8D                 5/09/88 19:44:33
OM_MECHANISMS.11.0.1D          11/16/88 13:16:27
(...)
```

Once the Environment has booted, log on and produce a summary of the error logs:

```
Io.Set_Output ("!Machine.Pm_>>Today's Date<<");
System_Report.Generate (Report_Type => System_Report.Everything,
                        Start_Time => ">>Last PM Date<<",
                        End_Time => "",
                        Log_Directory => "!Machine.Error_Logs");
```

This takes a few minutes. Examine the summary, specially checking the following items:

- Backups
  Make sure they do frequent backups.
- Memory ECC errors
  You may want to replace a board if there are numerous errors on a particular bit.
- Disk bad blocks
  Bad blocks may be an indication that a disk is getting sour.
- Exception messages
  They may indicate an abnormal behaviour of the Environment.

◆

INFO: Installing a 4th Fujitsu M2372K 823 Mb disk		
Applicable to:	*Fix:*	
References:	*Keywords:* Disk	
Originated from: Mv, Seh	*Revised by:*	

The purpose of this note is to describe the installation of a 4th Fujitsu M2372K 823 Mb disk drive. The switches on the disk drive should be in the following position:

```
SW1-1 to SW1-3: Logical Unit number:
Unit - Key 1, Key 2, Key 3
0       OFF     OFF     OFF
1       ON      OFF     OFF
2       OFF     ON      OFF
3       ON      ON      OFF <======

SW1-4: Tag 4/5 enable:
Enable:                 ON  <======

SW1-5: File protect:
Enable Writing      OFF   <======
Disable Writing     ON

SW1-6: Device type:
M2372K              OFF   <=======

SW1-7: Sector Mode
Hard Sector:            OFF <======
Variable Soft Sector: ON

SW2-1: Calibration Seek OFF
SW2-2: Not used
SW2-3: ON-Side switch:  OFF

SW3 - SW4: Sector Selection: ON = 1, OFF = 0
  SW3      SW4
1234567 1234567
1010110 0100000

SW5-1 to SW5-3: Spindle start delay switch
      1    2    3
delay 0: OFF  OFF  OFF

SW5-4: reserved
```

Once the disk is installed the recovery program should be run.

```
CLI> x recovery
Enter unit number of Disk to format/build: 3 [CR]
HDA : xxxxxxx
Disk appears intact. Should the information on it be used [Y] ? Y [CR]
```

```
Do you want to format the disk [N] ? N [CR]
Flagging bad blocks.
xxxx bad blocks flagged
Shall I perform a surface analysis [N] ? N [CR]

Do you want to specify new disk limits (WILL DESTROY DATA) [N] ? Y [CR]
Do you want to build a diagnostic file system on this unit [Y] ? N [CR]
Enter first cylinder to be used for read/write diagnositics: 740 [CR]
```

The recovery program must be run on unit 2 too, so that the system will not be able to boot, as the disk information is wrong. If needed for unit 0 a DFS must be built. The last cylinder of DFS is 37

Boot system and restore backup.

```
WANT TO BUILD NEW SYSTEM (YES/NO): yes [CR]
starting creation of new system
VOLUME NAME FOR unit 0: volume 1 [CR]
Want to set free blocks to gc footprint? no [CR]
VOLUME NAME FOR unit 1: volume 2 [CR]
VOLUME NAME FOR unit 3: volume 4 [CR]

Recovery Is needed, Should I fake it? no [CR]
Please tell me Vol Id for Backup Blue tape: .....
```

♦

INFO: DFS and R/W diagnostics limits	
Applicable to:	*Fix:*
References:	*Keywords:* Dfs, Disk
Originated from: Deg	*Revised by:*

When formatting a disk drive you will be asked to supply the first cylinder to be used for read/write diagnostics. If you also choose to build a DFS (Diagnostic File System) on that drive you will be prompted too for the last cylinder to be used by the DFS. The following table gives these values for the Fujitsu's drives.

```
Disk     DFS  R/W Diag
=====    ===  ========
F2351    50   835
F2361    50   835
F2333    72   815
F2344    37   620
F2372    37   740   -- formatted as a 2372, use 620 when formatted as a 2344.
```

The last cylinder to be used by the DFS is calculated as follows:

```
(20000 / ((H * S) / 2) + 1)
```

where:

H is the number of heads on the disk.
S is the number of sectors on the disk.
♦

INFO: How to reformat a disk drive		
Applicable to:	*Fix:*	
References:	*Keywords:* Dfs, Disk, Format	
Originated from: Deg, Gpa, Ken, Smp, Vnv		*Revised by:*

1. You should always have a complete full environment backup.
2. Or if you're one who believes "better to be paranoid than dead!" (GPA) then:
   - Do an incremental backup.
   - Do a full backup.
   - Archive stuff changed since the previous full.
3. If reformatting unit 0 (volume 1) then also have DFS backup. You can create this one by doing:

   ```
   CLI> backup
   ```

4. If the drive is not good, you may need a tape with the defects map; otherwise, the recovery program will read it from the drive.

   Then follow the procedure as described in the Product Installation Procedures Guide (87/12/15), Section 22.9, Reformatting Disk Drives, in summary:

   - Get to the CLI.
   - Execute the `Recovery` program.

     ```
     CLI> x recovery
     ```

   - Enter unit to build. If it isn't possible to read the bad block info, you will need the defects tape (one is shipped with each drive, or can be obtained from the home office, but you will need to specify the HDA number of the disk) here, otherwise, use the info on the disk.
   - Format the drive (10 minutes).
   - Perform 2 passes of surface analysis (10 minutes each pass).
   - Use [Break] to get out of cycle and reboot when done.

◆

TIP: Unibus parity error		
Applicable to:	*Fix:*	
References: CSR 3216	*Keywords:* Disk, Disk Tape Controller, Ecc Error, Unibus	
Originated from: Mbd, Rjg	*Revised by:*	

This is an interesting example of a case where a disk error message can be generated even when the disks, Disk/Tape Controller, and cabling are not at fault.

A machine hangs after getting disk errors. The messages look something like this:

```
Disk Error: Attempt (1023, DATA, 356, 11791) => (3, 11891)
Unit => 2
...
RMC52 Unibus Parity Error (UPE => True)
...
```

In this case, the problem turned out to be in the IOC. It happened that the IOC RAM is giving parity errors. The parity error is passed on to the UNIBUS, and the Disk Controller reports receiving them. That is how read parity errors are detected when the destination of the data is the UNIBUS.

♦

TIP: Swapping disks between machines	
Applicable to:	*Fix:*
References:	*Keywords:* Disk, Iop, Memory, Slew
Originated from: Lhrp	*Revised by:*

This card explains how to swap disks between two machines.

Prerequisites

1. Disks that will fit in each other's disk cabinet.
2. DFS on both sets of disks that contain the IOP kernel needed for the other machine. If not, the needed files will have to be read from a DFS tape.
3. If the disks have different numbers of sectors per track, a slew tape to slew the Disk/Tape Controllers.
4. Full verified backups in case slewing is involved.
5. Check which tapes you need at which stage, and check that the machines have the appropriate tape drives.

Swap

1. Bring machines down, power off.
2. Physically swap disks.
3. Power on machines.
4. Change `IOP ENVIRONMENT` configuration. You must set the configuration according to the type of machine that the disks came from. This is needed to get to the CLI prompt without hitting a problem with IOP kernel incompatibility when loading this kernel from the disk.
5. Slew the Disk/Tape Controller if the disks are not of the same type or are not formatted to the same size.
   - For compatible disks (only the number of cylinders differs, like M2344 and M2373) the machine can boot to the CLI; then you do:

     ```
     CLI> x slew
     ```

   - For incompatible disks a slew tape has to be used.
6. Change the IOP kernel that is loaded during the boot process. First type:

   ```
   CLI> dir *KERNEL_0*
   ```

   This should produce something like:

```
M200_KERNEL_0.M200
M300S_KERNEL_0.M200
M300C_KERNEL_0.M200
KERNEL_0.M200
```

The last is the one that is loaded to get to the CLI level. It is just a copy of one of the other three. Replace this file by the file needed for this machine. On a series 200 for instance:

```
CLI> copy M200_KERNEL_0.M200 KERNEL_0.M200
```

7. Using `cedit`, set the hardware configuration to reflect the correct memory boards on this machine.

8. At this point you may either:

   - Using `initioa`, set the Cluster Id to that of the machine the disks were taken from. This ensures that products will remain authorized, but is not recommended since we try to keep track of customer machines using the Cluster Id.

   - Keep the Cluster Id unchanged. In this case you will have to re-authorize all products once the Environment is started.

9. Crash the machine with the console break key.

10. Set the `IOP ENVIRONMENT` configuration back to the actual type of machine.

11. Reboot

Note:

- If the disks are compatible but the Disk/Tape Controller is slewed for a smaller number of cylinders, many error messages will appear during starting of the virtual memory:

```
Disk error (attempt 1023, data 259, 616) <=(1,658967)
  unit => 0, command => read
  RMER1: invalid_sector_position (IAE) => true
  RMDC: cylinder => 739
  RMDA: track => 0
  sector => 44
```

  This is because the software on the disk will ask to controller to access sectors that it ,the controller, doesn't know of.

- If the disks are compatible but the Disk/Tape Controller is slewed for a larger number of cylinders, no problems should occur.

♦

HINT: Fujitsu M244X tape drive density select		
Applicable to:	*Fix:*	
References: M244X Streaming Tape Drive Customer Engineering Manual		*Keywords:* Density, Tape
Originated from: Fujitsu	*Revised by:* Mv	

Normally at power up, density select is set to *host* changeable. For the R1000, that means that the tape will be written with the density with which the tape previously was initialized, unless the operator manually changes the density at the operator control panel.

However, it is possible to program the tape drive (although not in Ada) to always select 6250 BPI density.

From the operator control panel of the tape drive:

1. Push the `Start` key while keeping the `Test` key depressed. The "`Test`" LED lights and the 2 digit indicator displays "`00`".
2. Set the 2 digit indicator to "`93`" by pressing `Start` or `Unload`. `Start` causes the 2 digit indicator to increase and `Unload` decreases the indicator.
3. Push the `Density Select` key while keeping the `Test` key depressed. The 2 digit indicator displays "`dt`".
4. Push the `Reset` key. In this state, the current device-type code is displayed on the 2 digit indicator. (The code is usually "`C0`").
5. Use `Start` and `Unload` to alter the value to "`C2`".[1]
6. Push the `Test` key. The displayed device-type code is stored into RAM, the 2 digit indicator displays "`EL`".
7. Push the `Reset` key. Push the `Test` key. The 2 digit indicator displays "`bG`".
8. Push the `Reset` key. Push the `Test` key. The 2 digit indicator displays "`EG`".
9. Push the `Reset` key. Push the `Test` key. The 2 digit indicator displays "`00`".
10. To store the new parameters into nonvolatile memory, push the `Start` key to set "`94`" (Store parameter command) and push the `Density Select` key while holding down the `Test` key. The 2 digit indicator displays "`00`".
11. Push the `Reset` key to exit test mode.

◆

[1] altering the value to "`82`" will also inhibit streaming mode

INFO: Tape drive - Retry_Count_Exhausted	
Applicable to:	*Fix:*
References:	*Keywords:* Tape, Tape Error
Originated from: Deg	*Revised by:*

A tape can be thought of as a bunch of very small magnets grouped into blocks. There are nine rows (channels) of these magnets, the number of columns defines the number of characters in a frame where one column equals one character. A tape head mechanism is made up of nine separate heads, one for each channel. The head can create a magnetic field on the tape (write) and read the orientation of the magnetic field from the tape (read). Sometimes these magnetic fields on the tape are so small a head has a hard time reading it. This can be caused by many reasons:

- Dirty head (clean it well),
- Tapes inability to hold a magnetic charge (bad tape),
- Tape drive read or write gain is not set right (sometimes can adjust),
- Cables from controller to the drive is bad or has a loose connection,
- Problem with the Disk/Tape Controller.

Sometimes, the heads are not aligned correctly (called *skew*). One drive may have no problem reading and writing it's own tapes but cannot either produce a tape readable by another drive, or read a tape produced on another drive. This type of adjustment is not easy to correct (by Rational).

Fortunately, there are many error correction schemes used by tape drives to compensate for some of these problems. But sometimes bad data is not correctable. When this happens, errors are reported to the Disk/Tape Controller which results in messages entered in the error log. The drive attempts to correct the data 12 times before it fails.

Overall, when writing tapes, keep eyes open for write errors as this can be a sign of a bad tape being produced. Heads should be kept clean and good tapes used. If one particular drive appears to be sensitive to reading tapes made on another drive, or other drives have a problem reading tapes made from a particular drive, the gain may need to be adjusted or the drive to be skewed.

◆

INFO: Kennedy tape drive - preventive maintenance	
Applicable to:	*Fix:*
References:	*Keywords:* Kennedy, Preventive Maintenance, Tape Drive
Originated from: Deg	*Revised by:*

You will need the following:

- A Phillips screw driver.
- A head cleaner (isopropyl alcohol).
- A lint free cloth and water.

These are the steps:

- Slide out Kennedy from the bay and open the top of the tape drive by loosening the two screws on the lid. Pop open cover covering the heads.
- With the head cleaner, clean the head, flux gate (holds tape against head, will be retracted), then the six tape guides. Do not use head cleaner (isopropyl alcohol) on rubber, i.e. the capstan, as it causes rubber to crack.
- With a lint free cloth and water (mild soap if really dirty), clean the capstan roller, then wipe down tape channel and cabinet.
- Run diagnostics.

Note: No routine adjustments are necessary and *should not* be made except as a corrective action.
♦

TIP: Kennedy tape drive and tape loading	
Applicable to:	*Fix:*
References:	*Keywords:* Kennedy, Tape, Tape Drive
Originated from: Deg	*Revised by:*

1. Not loading tapes.

   - It appears that the Kennedy tape drive has a loading problem with smaller tapes. The supply reel lets out just enough tape to wrap around and grip the take-up reel. In the case of smaller tapes, less tape is let out to wrap around and grip the take-up reel. Because we are using graphite backed tapes (to aid in slipping over rollers) the tape also slips off the take-up reel causing the loading problem.

     Making the tape less slippery in the first 10" of tape seems to fix the problem. We were able to accomplish this by rubbing 10" of the back of the tape with our fingers. The grease on our fingers made the graphite surface less slippery. Using a larger tape or a non graphite backed tape seems to avoid this problem also.

   - Static build up can cause the tape not to load. On some new tapes, a piece of tape hold the tape tight. Upon peeling the tape off static electricity it produces causing the beginning of the tape to stick to the reel. When loading, the beginning of the tape must flap in front of a sensor 3 times before it will attempt to load the tape. If after so many tries, the tape has not passed in front of the sensor 3 times then the load will fail.

     Lightly crimp the beginning of the tape along the center (only about a 1/2" to 1"). This will help the tape spin of the supply reel. If this is suspected of being a problem just view that the tape is not stuck onto the reel when loading.

2. Long loading time.
   The Kennedy will attempt to auto-load a tape 3 times before it reports a failure. If the tape seems to take an extraordinary long time to load a tape, it may be because it successfully loaded on the second or third attempt.

   This could be related to the graphite backed tapes. If not, diagnostics will need to be run to isolate the problem. First check to make sure that the tape is flowing down the tape channel properly. This can be done by attempting to load the tape until the supply reel starts to rewind. Then turn off the tape drive, open up the top, and view that the tape is wrapped around the take-up reel.

◆

INFO: Kennedy tape drive error messages	
Applicable to:	*Fix:*
References:	*Keywords:* Kennedy, Tape Drive
Originated from: Pam	*Revised by:*

1. Power up

Upon power up, and without a reel of tape inserted, the readout normally displays: TESTING, for approximately 7 seconds. If TESTING continues to be displayed, this indicates that communication between formatter and drive is disrupted and no further testing can be accomplished. There is a problem in the formatter or the interface between formatter and drive. Assuming that communication between formatter and drive is obtained, the drive performs the self-tests procedure. In addition to the failure codes, the drive can display failures in the form of words or abbreviations, and also indicate the possible causes of failure.

The following tables display the error messages and related explanations.

DISPLAY	FAULT CONDITION/POSSIBLE FAILURE
TESTING	**Formatter not communicating with the drive**: Diagnostics hangup: formatter board 8951, drive electronic board 8952
FMT ERR? and flash REW	**Formatter error (formatter timeout)**: formatter section has failed to interrogate drive electronics; or the formatter has failed to send its identification (ID) after request from the drive electronics. Check formatter board 8951.
TAK ARM?	Take-up arm fails to calibrate.
SUP ARM?	Supply arm fails to calibrate.
CAPSTAN?	No tachometer voltage. Check capstan motor, capstan power amplifier, and tachometer.
T MOTOR?	**Take-up reel motor not running**: Servo preamplifier 7147, servo amplifier 7111, drive electronics board 8952, take-up reel motor.
S MOTOR?	**Supply reel motor not running**: Servo preamplifier 7147, servo amplifier 7111, drive electronics board 8952, supply reel motor.
ROM ?	**Drive electronics program PROM checksum error**: Non-zero checksum detected.
EEPROM ? and	**Checksum error**: Non-zero checksum detected. Press RWND/UNL. The

drive		
flash REW	compute a new checksum and attempt to write it into the EEPROM. Check drive electronics board 8952.	
+12V ?	Voltage below +10V	**Check applicable**
+40V ?	Voltage below +30V	**power supply**
-12V ?	Voltage above -7V	**and regulator**
-40V ?	Voltage above -29V	
FAIL XXX	(XXX refers to failure code - See table Below)	

The following table lists the failure codes that could display during power up, indicating probable causes of failure. Note that failure codes 150 through 159 include the option of obtaining a second failure code which indicates specific problems. To obtain that second failure code, press **Density**.
Note: The failure code is preceded by the word: FAIL.

CODE	DESCRIPTION	PROBABLE FAILURE
128	External RAM data error	Compare RAM formatter 8951
129	External RAM addressing error	
131	No RSBY for read amp gain value	Read analog board 9017
132	Wrong formatter board	Formatter board 8951
133	Wrong analog board	Read analog board 9017
134	Wrong digital board	Read digital board 9060
140	Auto adjust ranging error	
141	Dead track found in auto adjust	Read analog board
142	Over range signal in auto adjust	
150	No RSBY in loop RAW test	
151	No WSBY in loop RAW test	
152	Time-out on WSBY in loop RAW test	
153	Time-out on RSBY in loop RAW test	Press **DENSITY**
154	WSBY ended to soon in loop RAW test	pushbutton for
155	Hard error in loop RAW test	second code
156	Corrected error in loop RAW test	listed below
157	Compare error in loop RAW test	
158	File mark during loop RAW test	
159	Block size error in loop RAW test	

ITEM	2ND CODE	DESCRIPTION	PROBABLE FAILURE
1	D-50 NRZ	LWRD, 50 ips NRZI	
2	D-100 NRZ	LWRD, 100 ips NRZI	
3	D-50 PE	LWRD, 50 ips PE	
4	D-100 PE	LWRD, 100 ips PE	Read digital
5	D-50 DPE	LWRD, 50 ips DDPE	board 9060
6	D-100 DPE	LWRD, 100 ips DDPE	
7	D-50 GCR	LWRD, 50 ips GCR	
8	D-100 GCR	LWRD, 100 ips GCR	
9	A-100 PE	LWRA, 100 ips PE	
10	A- 50 DPE	LWRA, 50 ips DPE	Read analog 9017
11	A-100 DPE	LWRA, 100 ips DPE	
12	A- 50 GCR	LWRA, 50 ips GCR	

Legend:

```
RSBY = Read Busy Signal
WSBY = Write Busy Signal
RAW  = Read After Write
LWRD = Loop Write to Read Digital
LWRA = Loop Write to Read Analog
```

2. **Loading failure codes**

The tape drive also includes automatic diagnostics that can detect loading failures, giving appropriate displays to indicate failures and their possible causes.

DISPLAY	FAULT CONDITION/POSSIBLE FAILURE
COVER?	**Door interlock circuit**: (drive cannot load) Tape access door open, door switch, drive electronics board 8952, interconnect board 7228.
NO BOT?	**No BOT marker detected**: Tape marker missing from tape, tape leader too short, faulty BOT sensor, drive electronics board 8952, interconnect board 7228.
ABORTED	**Automatic load aborted**: No vacuum, air leak in tape path, drive electronics board 8952, servo boards 7111, 7147, position sensors, tape sticking.
REV REEL	**Reverse reel - Reel upside down.**
PLC REEL	**Place Reel.** Tape reel not installed.
CHK SLND	**Check solenoid - hub lock solenoid not locking**: Hub lock solenoid defective or needs adjustment, drive electronics board 8952, interconnect board 7228.
CHK HUB	**Check hub - Supply reel not detected**: Reel-in-place tab adjustment, reel-in-place sensor, drive electronics board 8952, interconnect board 7228.
BKN TAPE	**Broken tape - Tape not detected**: Broken tape, no EOT marker, EOT sensor, interconnect board 7228, drive electronics board 8952.
TAK ARM?	**Take-up arm not operating correctly**: Take-up arm position sensors (Check with arm relaxed), drive electronics board 7710, interconnect board 7228.
SUP ARM?	**Supply arm not operating correctly**: Supply arm position sensors (Check with arm relaxed), drive electronics board 7710, interconnect board 7228.

◆

24. Exit data diagnostics by pressing `Diag` until `UNIT 0` is displayed.

◆

Legend:

```
RSBY = Read Busy Signal
WSBY = Write Busy Signal
RAW  = Read After Write
LWRD = Loop Write to Read Digital
LWRA = Loop Write to Read Analog
```

2. **Loading failure codes**

The tape drive also includes automatic diagnostics that can detect loading failures, giving appropriate displays to indicate failures and their possible causes.

DISPLAY	FAULT CONDITION/POSSIBLE FAILURE
COVER?	**Door interlock circuit**: (drive cannot load) Tape access door open, door switch, drive electronics board 8952, interconnect board 7228.
NO BOT?	**No BOT marker detected**: Tape marker missing from tape, tape leader too short, faulty BOT sensor, drive electronics board 8952, interconnect board 7228.
ABORTED	**Automatic load aborted**: No vacuum, air leak in tape path, drive electronics board 8952, servo boards 7111, 7147, position sensors, tape sticking.
REV REEL	**Reverse reel - Reel upside down.**
PLC REEL	**Place Reel.** Tape reel not installed.
CHK SLND	**Check solenoid - hub lock solenoid not locking**: Hub lock solenoid defective or needs adjustment, drive electronics board 8952, interconnect board 7228.
CHK HUB	**Check hub - Supply reel not detected**: Reel-in-place tab adjustment, reel-in-place sensor, drive electronics board 8952, interconnect board 7228.
BKN TAPE	**Broken tape - Tape not detected**: Broken tape, no EOT marker, EOT sensor, interconnect board 7228, drive electronics board 8952.
TAK ARM?	**Take-up arm not operating correctly**: Take-up arm position sensors (Check with arm relaxed), drive electronics board 7710, interconnect board 7228.
SUP ARM?	**Supply arm not operating correctly**: Supply arm position sensors (Check with arm relaxed), drive electronics board 7710, interconnect board 7228.

◆

INFO: Kennedy tape drive - data read/write diagnostics	
Applicable to:	*Fix:*
References:	*Keywords:* Kennedy, Tape Drive
Originated from: Deg	*Revised by:*

1. Load a write enabled scratch tape.
2. Enter the diagnostic mode by pressing `Diag`. `DIAG` is displayed.
3. Press `Enter`. `SELFTEST` is displayed.
4. Press `Scan` until `DATADIAG` is displayed.
5. Press `Enter`. `MODE` is displayed.
6. Press `Enter`. `ON ERROR` is displayed.
7. Press `Enter`.
8. Press `Scan` until `CONTINUE` is displayed.
9. Press `Enter`. `AT EOT` is displayed.
10. Press `Enter`.
11. Press `Scan` until `STOP` is displayed.
12. Press `Enter`.
13. Press `Diag`. `MODE` is displayed.
14. Press `Scan` until `COMMANDS` is displayed.
15. Press `Enter`. `LOOP DIG` is displayed.
16. Press `Scan` until `WR/RR/RD` is displayed.
17. Press `Enter`.
18. Start the test by pressing `Start/Stop`. Diag light flashes.
19. After about 15 minutes stop testing by pressing `Start/Stop`.
20. Press `Diag`. `COMMANDS` is displayed.
21. Press `Scan` until `ERRORS` is displayed.
22. Press `Enter`. `EIC0 n` is displayed (n errors in channel 0).
23. Press `Scan` to retrieve other errors until `ERRORS` is displayed. Record the error results.

24. Exit data diagnostics by pressing `Diag` until `UNIT 0` is displayed.

◆

INFO: Kennedy tape drive - how to unlock a tape reel	
Applicable to:	*Fix:*
References:	*Keywords:* Kennedy, Tape, Tape Drive
Originated from: Vnv	*Revised by:*

The reel locking fingers can be manually locked or unlocked by depressing the manual reel locking lever while turning the supply hub. Then manual reel locking lever is located inside the front access door on the bottom to the left.

If a tape is stuck on the supply hub and unloading doesn't seem to work, turn the power on (this makes it easier to turn the take-up reel if the tape is wound on it) and then spin the supply reel until the tape is completely wound onto the supply reel (hope you're near the BOT). Then depress the manual reel locking lever and turn the supply hub counter clockwise until the reel locking fingers unlock.

♦

INFO: Tape drive switch	
Applicable to:	*Fix:*
References:	*Keywords:* Tape Drive, Tape Drive Switch
Originated from: Hts, Smp	*Revised by:*

The recommended setup is to plug CPU0 controller cables into the tape drive switch CPU0 connectors on the tape drive next to CPU 0, and CPU1 controller connectors into the CPU1 connector. Use the tape switch key in the position relating to the CPU (0 or 1).

The EXPAND outputs are not yet implemented (the terminators currently installed are not correct) on the tape drive switch, thus do *not* connect anything into these connectors.

If no tape drive switch is available a workaround is to bypass the switch altogether as follows:

If CPU 0 requires the tape drive connect cable marked CPU0-A to cable marked TAPE DRIVE-A and cable marked CPU0-B to cable marked TAPE DRIVE-B.

If CPU 1 requires the tape drive connect cable marked CPU1-A to cable marked TAPE DRIVE-A and cable marked CPU1-B to cable marked TAPE DRIVE-B.

◆

PROBLEM: Potential Tape Drive failure due to simultaneous access	
Applicable to: D_10_20_2, D2	*Fix:*
References:	*Keywords:* Exabyte, Kernel, Tape, Tape Drive
Originated from: Mam, Vnv	*Revised by:*

For systems that have both a 9-track and an 8-mm cartridge tape drive, simultaneous access to both tape drives can cause both drives to fail when either of the drives is used by a program that selects a block size of 4K. Such programs include:

- user-written programs,
- `Tape.Read` and `Tape.Write` operations that specify a block size of 4K.

You *can* use both tape drives concurrently for any other combination of tape operations, including `Archive`, backup, TBU, user-written programs that select a block size of 3K or less.

This is because both tape drives will together request too many IOP buffers and this will cause the machine to crash with a `Kernel Assert Failure`.

◆

PROBLEM: Potential Tape Drive failure due to

Rev. 0

INFO: Troubleshooting Exabyte problems	
Applicable to:	*Fix:*
References:	*Keywords:* Exabyte, Tape Drive
Originated from: Mac	*Revised by:*

The problem is that the door on the drive would not open by pressing the button on the front of the drive: you never even get as far as being able to insert a tape. At power up, both the green and amber lights come on momentarily, then the green light goes out and the amber one stays on.

In the EXB-8200 CTS Product Specification is the following table for power pin assignments for the power connector:

PIN NO.	USAGE
1	+12V
2	Ground
3	Ground
4	+5V

Checking the voltages on the power cord off the Exabyte power card, shows that +5V is fine, but no +12V is measurable: it powers the servos to open the door on the drive.

The solution is to replace the power card.

♦

INFO: M2372K disk drive - Disk/Tape Controller slewing	
Applicable to:	*Fix:*
References:	*Keywords:* Board, Disk, Disk Tape Controller, Slew
Originated from: Mv, Seh	*Revised by:*

The purpose of this note is to give the parameters for slewing the Spectra Logic disk/tape controller for a Fujitsu M2372K 823 Mb disk. If the `Slew` does not specifically support the 2372, then the `Other` function must be used.

```
Tag 4/5 enabled => YES
Transfer rate exceeds 2MB => YES
Number of cylinders => 745
Number of heads => 27
Number of sectors => 66
```

◆

INFO: Unofficial policy regarding memory ECC errors		
Applicable to:	*Fix:*	
References:	*Keywords:* Crash, Ecc Error, Memory	
Originated from: Dbp, Rjg, Sjf		*Revised by:*

Naturally, all this depends somewhat on the political situation with the customer, but here are some guidelines:

Occasional ECC errors are to be expected, this is why the error correcting mechanism is in place. Multi-bit errors, however, are cause for alarm (the machine will crash) and should be dealt with.

30+ ECC errors per day on the same board (especially if on the same bit) are cause for concern, and a board replacement should scheduled with the next PM.

100+ ECC errors per day are cause for scheduling board replacement at the next convenient time.

1000+ ECC errors per day are cause for alarm because of the increased likelihood of multibit errors, as well as the performance cost of error correction. The board should be replaced ASAP.

♦

TIP: Multi-bit memory error	
Applicable to:	*Fix:*
References:	*Keywords:* Cli, Crash, Ecc Error, Memory
Originated from: Mam, Seh	*Revised by:*

The purpose of this card is to be able to find out when the system crashes after a multi-bit error which memory board had this error.

1. Do not run the FRUs, or hit the break key or white button.

```
Run the frus [Y] : N
boot [Y] : N
Cli>
```

2. Enter the CLI.

3. Run the Experimental Monitor EXPMON

```
cli> x expmon
EM>
```

4. Run the Who_Hit program on each board. This program is invoked by the X? command where the ? is one of J, K, L, M the names of the boards. The board that comes back with a TRUE is the bad board.

```
EM> XJ WHO_HIT

    FALSE FALSE

EM> XK WHO_HIT

    TRUE FALSE
```

...this one is the bad board

```
EM> XL WHO_HIT

    FALSE FALSE

EM> XM WHO_HIT

    FALSE FALSE
```

Note: J = MEM3, K = MEM2, L = MEM1, M = MEM0.

♦

INFO: Memory and disk errors	
Applicable to:	*Fix:*
References:	*Keywords:* Disk, Ecc Error, Memory
Originated from: Deg	*Revised by:*

Memory ECC errors are natural errors that occur from time to time on memory boards. Single bit errors are correctable using ECC (error correction code) but multi bit errors are a sign of a bad memory board and can crash the machine. Typically a memory board can receive 1000's of ECC errors before a multi bit error might occur. If over a 100 ECC errors occur in less than a week then this is a sign of a memory board going bad and that should be scheduled to be replaced.

Disk errors fall into two categories. Read errors and write errors. Write errors are typically not that critical since they are usually retargeted and the bad block marked. Read errors on the other hand can be very damaging. If a problem reading data is encountered the drive retries reading the data 10 times. If not successfull the heads are shifted to either side and read another 10 times. After a total of 30 read attempts occur the drive produces a disk error. If during the retries the data is recovered it is then retargeted. If there are more than 5 retargeted bad blocks then this may be a sign a bad disk. Running surface analysis may fix the bad blocks. If disk errors appear to be on same track then there may be a bad head.

◆

INFO: Spare Disk/Tape controller slew procedure	
Applicable to:	*Fix:*
References:	*Keywords:* Board, Cli, Disk, Slew, Spare, Tape
Originated from: Deg	*Revised by:*

Disk/Tape Controllers sent out as spares may be initially slewed as follows:

```
Unit 0 - Eagles
Unit 1 - XP's
Unit 2 - 2333 8"
Unit 3 - 2344 8"
```

This provides the ability to slew the controller from the DFS on disk in the event that a slew tape is not available. If a slew tape is available then using it is probably the simpler way to go but if not then follow the direction below.

- Write protect and power off disk drives.
- Turn DC power off and install Disk/Tape Controller.
- Set the unit number on the disk drive with the DFS to match the unit number for the drive type on the Disk/Tape Controller (See above).
- Turn only this disk drive ON. Unwrite protect drive when READY.
- Turn DC power on and *white button* the machine.
- Open dip 4 on switch 1 of the Disk/Tape controller.
  In front of you when you look in the cardcage at the Disk/Tape Controller. (same position as dip 2, opposite dip 1,3).
- Boot to `CLI`.
- Run slew program.

  ```
  CLI> X SLEW
  ```

  Complete slew procedure.
- Close dip 4 on switch 1.
- Write protect and power OFF disk drive.
- Set the unit number on the disk drive with the DFS back to the original unit number.
- *White button* the machine and you are done.

♦

INFO: VAL board P2ucode failed with the Val_Last condition	
Applicable to:	*Fix:*
References:	*Keywords:* Board, Cli, Crash, Diagnostic, Frus, Microcode
Originated from: Vnv	*Revised by:*

```
Value board P2ucode failed
Problem was detected with the Val_Last condition
Field replaceable units Value Board
```

The FRUs are incomplete in this area, and in general if replacing the VAL board does not fix it, the SEQ is the next most likely board, followed by any other R1000 board.

♦

INFO: VAL board P2ucode failed with the Val_Last

Rev. 0

INFO: Board has a bad WCS bit		
Applicable to: SEQ, VAL, IOC boards		*Fix:*
References:	*Keywords:* Board, Crash	
Originated from: Vnv	*Revised by:*	

The machine crashes and reboots with the message:

```
Loading Control Store ..............  [OK]

XXX board has bad WCS bit at address YYYY.
The faulty chip is at location ZZZ.
Reloading the control store fixed the ram location(s).
CLI>
```

The FRUs won't catch anything, the machine will be able to reboot, but the problem will recur. The board should be changed as soon as possible. Note down board, address, and chip location (if given) at time of crash (on the operator console).

♦

INFO: Processor refresh machine check crash message	
Applicable to:	*Fix:*
References:	*Keywords:* Board, Crash, Microcode
Originated from: Vnv	*Revised by:*

The machine crashes displaying:

```
restarting r1000 after processor refresh machine check
...
Delta:
Mom:
Cluster Manager Error at PC=3ab8, Error Code = 0C
Cluster Manager detected R1000 machine check
...
```

`Processor refresh machine check` refers to the following:

there are two counters in the FIU that keep track of when memory was last refreshed (since the memory is dynamic, it must be occasionally refreshed). If the request to refresh memory is ignored too long, the machine crashes with this message. Usually this indicates a problem with the FIU board or the microcode.

◆

TIP: FPTEST fails with Arithmetic_Error	
Applicable to:	*Fix:*
References:	*Keywords:* Arithmetic Error, Board, Cli, Exception, Frus, Val
Originated from: Unknown	*Revised by:*

When running:

```
CLI> FPTEST
```

it fails immediately with:

```
Read_Novram_Data.VCal abort 64 bit arithmetic error
words error;9
from novram
Read_Novram_Data.VCal abort 64 bit arithmetic error
words error;9
from configure
....
```

Replacing the VAL board fixes the problem.

◆

TIP: EEPROM error on Disk/Tape controller	
Applicable to:	*Fix:*
References:	*Keywords:* Board, Boot, Disk Tape Controller, Eeprom, Flow Control, Pcm, Pdu, Slew
Originated from: Ken	*Revised by:*

During the boot process, just after the disks online messages, the following is displayed:

```
Warning -- EEPROM checksum error, disk/tape controller 0
```

The system may go ahead and boot.

Note: The above warning message does not show up in the machine error logs.

This is usually caused by powering the machine off with the switch 4 on the tape disk controller open thus allowing the EEPROM to get clobbered however it has been seen failing even with the switch closed.

The fix is to run the slew procedure to *repair* the EEPROM checksum error.
◆

INFO: Excelan controller error codes	
Applicable to:	*Fix:*
References:	*Keywords:* Board, Ethernet, Excelan, Exos, Network, Tcp Ip
Originated from: Jmk, Pam, Tdg	*Revised by:*

1. **Self-diagnostics and configuration errors**

   - Introduction

     The built-in tests can isolate specific software or hardware errors associated with configuration or operations. The errors are reported via a LED making it possible for them to be identified and corrected.

   - LEDs

     There are four LEDs along the visible edge of the Excaln board. Three of the LEDs are in a group located about 5 cm below the cable connector. Bottom to top, these three LEDs are labeled DS1, DS2, and DS3; the labels cannot be seen while the board is installed, but the names will be useful in describing the LEDs. The last LED, DS4, is located near the bottom of the board, partly hidden by the board's ejection lever.

     - DS1

       If DS1 is on or pulsing, it indicates an error. Error codes are 8-bit numbers, and are presented bit-by-bit, starting with the most significant bit. The error code is continuously repeated, with a pause in between to demarcate the starting point.

     - DS2

       DS2 indicates Ethernet activity.

     - DS3 and DS4

       DS3 and DS4 indicate Unibus activity.

   - Error categories

     The errors reported by DS1 are logically categorized into three groups.

     - 16#A0# .. 16#AF#: Fatal software configuration errors
     - 16#B0# .. 16#BF#: fatal hardware errors
     - 16#C0# .. 16#CF#: Fatal errors, either software or harware, which occur during the course of normal operation.

   - Configuration errors codes

     ```
     HEX Code     Pulse Code     Explanation of Error Code
     ```

16#A0#	-.-.	Invalid address for configuration message
16#A4#	-.-. .-..	Invalid operation mode parameter
16#A5#	-.-. .-.-	Invalid host data format pattern
16#A7#	-.-. .---	Invalid configuration message format
16#A8#	-.-. -...	Invalid movable data block parameter
16#A9#	-.-. -..-	Invalid number of processes parameter
16#AA#	-.-. -.-.	Invalid number of mailboxes parameter
16#AB#	-.-. -.--	Invalid number of address slots parameter
16#AC#	-.-. --..	Invalid number of hosts parameter
16#AD#	-.-. --.-	Invalid host queue parameter
16#AE#	-.-. ---.	Improper objects allocation
16#AF#	-.-. ----	Net boot failed
16#B0#	-.--	Checksum on NX 200 EPROM failed
16#B1#	-.-- ...-	Memory test failed for 0-128k
16#B2#	-.-- ..-.	Memory test failed for 128K up to the highest address
16#B3#	-.-- ..--	Counter test failed
16#B4#	-.-- .-..	Interrupts test failed
16#B5#	-.-- .-.-	Transmission test failed
16#B6#	-.-- .--.	Receive test failed
16#B7#	-.-- .---	Local loooback data path test failed
16#B8#	-.-- -...	CRC test failed
16#B9#	-.-- -..-	Checksum on physical address EPROM failed
16#BA#	-.-- -.-.	System error
16#BB#	-.-- -.--	Ethernet chip initialization failed
16#BC#	-.-- --..	Ethernet chip self-test failed
16#BD#	-.-- --.-	Ethernet chip ressource counter failed
16#BE#	-.-- ---.	External loop-back test alignment error
16#BF#	-.-- ----	SBX board not in place
16#C0#	--..	Specified time exhausted
16#C1#	--.. ...-	Host memory read/write test failed
16#C8#	--.. -...	Parity hardware logic failed
16#C9#	--.. -..-	NMI interrupt for bus timeout failed
16#CA#	--.. -.-.	Host interrupt test failed
16#CB#	--.. -.--	Command unit test failed
16#CC#	--.. --..	Divide error exception
16#CD#	--.. --.-	Undefined interrupt type
16#CE#	--.. ---.	Command not executed by the CU of the 82586
16#CF#	--.. ----	Command block sync failed between hw and sw

2. Console error messages

Two types of information appear on the system console. When the EXOS front-end board is initialized, it reports a variety of configuration and version information including the software/firmware/hardware versions and the Internet and Ethernet addresses the networking software thinks are its own.
The second category of console messages consists of warnings that occur during system operation. The following messages can appear; they provide clues to various trouble situations.

- **EXOS CODE 0001**
  This occurs when the TCP checksum calculated by the EXOS board for an incomming packet does not agree with the checksum in the packet header. Check the intercommunicability of various hosts with each other to determine which is malfunctionning.

- **EXOS CODE 0003 rxmt time = nnn**
  This indicates that the front-end was forced to retransmit packets due to lack of response. It is normal for this to happen with *nnn* = 2 the first time you connect with a host. Trouble is indicated if it happens repeatedly with increasing values of *nnn*. The most likely causes are the following:

  - The host with which communication is being attempted is down or is unable to communicate. Check if that host can communicate with another system.

  - The IP address of the remote host is not properly entered in the `Transport_Name_Map`. You should ensure that both hosts agree on what each others' Internet addresses are. Check the following places for consistency: the `Transport_Name_Map` of the two machines and the addresses that the networking software believes to be their own.

- **EXOS CODE 0102 xmit err 10**
  This message appears when the EXOS board is having trouble transmitting. This typically happens because the transceiver is not properly connected to the Ethernet cable. This causes excessive collisions.
  Another possible cause is a general problem with the network, such as a short circuit.

- **EXOS CODE 0102 xmit err 04**
  This message indicates DMA underrun for the Ethernet chip. It can be reported by NX revisions greater than 4.7. If this problem persists, report it to Excelan.

- **EXOS CODE 0102 xmit err 20**
  This message indicates a problem in the attachment of the local node to the Ethernet cable. The symptom is lack of carrier sense during transmission.

- **EXOS CODE 0102 xmit err CB**
  This message results from one of two causes. Either the attachement to the network is faulty (as in the preceding paragraph) or there is a hardware fault on the front-end processor. The symptom is a timeout during transmission.

- **EXOS CODE 0104 D4**
  This is an error message that is not documented, but has been infrequently seen. It is followed by panic messages such as:

```
14:39:01 --- Ethernet Controller_Status EXOS STACK TRACE:
             3463,CEA,135C,8FEA,67B8,7494,8547,1
14:39:01 --- Ethernet Controller_Status EXOS PANIC MODE
```

  Each time this error has been seen, it was possible to recover by restarting the Ethernet controller.

- **EXOS CODE 0105 rcvd err 04**
  This indicates a DMA overrun within the front-end configuration. If it happens persistently and interferes with system operation, report the problem to Excelan.

- **EXOS CODE 0105 rcvd err 10**
  A packet longer than the specification permits was received.

- **EXOS CODE 0106** and **EXOS CODE 0107**
  An inconsistent constructed packet was received. UNIX 4.2BSD networking produces trailer packets. This code indicates that they are being produced incorrectly. Pursue the problem with the system(s) that might be transmitting these packets.

- **EXOS CODE 0115**
  No route exists to a correspondent.

- **EXOS CODE 0207, EXOS CODE 0208** and **EXOS CODE 0616**
  This message indicates a shortage of data buffers on the front-end processor. This may occur if any connection are actively transferring data concurrently, especially if the host stops reading data for numerous connections, forcing data to accumulate on the front-end. If this situation persists in normal connections, add memory or reduce the number of active connections. In any event, arrange for host applications to read input data from the board most of the time.

- **EXOS CODE 0301** *parameters*
  A host socket request contained illegal parameters such that it failed to specify a protocol that EXOS supports. *parameters* are displayed in hexadecimal in the following order:

  - Type of protocol within a protocol family
  - Protocol family
  - Protocol within family
  - Socket options
  - Internet protocol family
  - Internet TCP/UDP port
  - Internet socket address

  To find the source of the problem, investigate application programs attempting to open sockets.

- **EXOS CODE 0408 nnn** The EXOS software does not recognize a request code received from the host. There is probably a bug in the host's driver software. The number *nnn* is the improper request code. To find the source of the problem, investigate application programs trying new things. If you are using only Excelan-provided software, report this error to Excelan as a driver utility problem.

- **EXOS CODE 0705** and **EXOS CODE 0706**
  These are checksum calculation mismatches similar to EXOS CODE 0001. Code 0705 pertains to the ICMP protocol, code 0706 to IP.

- **EXOS CODE xxxx**
  If code that are not listed above are displayed and the EXOS is not working correctly, report the problem to Excelan.

3. **Ethernet controller error code after rebooting**

   After a white button reboot, the DS1 LED is always pulsing the sequence 16#BA# (`-.---.-.`). All the other LEDs may be off. The EXOS 204 manual lists this as an ambiguous system error.

The IOP diagnostics leave it in this state and it isn't cleared until `Tcp_Ip_Boot` is run by `!Machine.Initialize_Network` (after the Environment is booted).

Note: In case of problem during operation, to get a meaningful reading from DS1 you have to look at it before crashing the system.

◆

BUG: Ethernet controller status EXOS PANIC MODE	
Applicable to: D1	*Fix:* D2
References:	*Keywords:* Board, Ethernet, Excelan, Exos, Network
Originated from: Jmk	*Revised by:*

Excelan says they have fixed a software problem that resulted in the Ethernet controller crashing with error messages like this:

```
09:42:43 --- Ethernet Controller_Status EXOS CODE 020A
09:42:43 --- Ethernet Controller_Status EXOS STACK TRACE: ...
09:42:53 --- Ethernet Controller_Status EXOS PANIC MODE
```

The fix will be released in Delta 2. In the mean time, it can be retrofitted into a Delta 1 system, by copying (using an `Archive` tape, for example) three files into `!Tools.Networking`, and then restarting the Ethernet controller. The three files are:

```
!Tools.Networking.Exos_8000_4_Aa
!Tools.Networking.Exos_8000_4_Aa_Backup
!Tools.Networking.Exos_8000_4_Hb
```

You will find these files on **Clem**. I'd recommend you make an `Archive` tape containing them, and ship the tape out to the customer.

♦

INFO: SYS board error message on series 200/300	
Applicable to:	*Fix:*
References:	*Keywords:* Board, Sys Board
Originated from: Unknown	*Revised by:*

On model 200s/300s the IOA and Sysbus are combined into the IOC board, but messages still say `Sys_Board`. So for model 200s/300s this means the IOC. Which is in the `Sys_Board` slot in the board cage.

♦

INFO: IO processor kernel error	
Applicable to:	*Fix:*
References:	*Keywords:* Debugger, Iop, Kernel
Originated from: Ken	*Revised by:*

The crash message looks like this:

```
Io Processor kernel error 0601 at PC=00004D42, trapped into debugger.

RD0=0000FFFF, RD1=00000003, RD2=0000000F, RD3=D3D3D3D3,
RD4=00000000, RD5=D5D5D5D5, RD6=D6D6D6D6, RD7=D7D7D7D7,
RA0=00007480, RA1=9302E050, RA2=9303F160, RA3=00007480,
RA4=A4A4A4A4, RA5=00007690, RA6=A6A6A6A6, ISP=00005A14,
PC=00004B14,  USP=0003FFFC, MSP=924551C9, SR=2700,
ZBR=00000000,
ICCR=00000001, ICAR=DA6360A0, XSFC=5, XBFC=1

@
```

The @ prompt indicates you are at the debugger.

1. Write down the error message and the register dump.
2. Look at the ISP value in the register dump, 00005A14 in the example, and at the debugger type in:

   ```
   00005A14,40/
   ```

   This displays 40 long (32 bits) data word starting at address 00005A14. Write down what you get, then continue.
3. Use [Esc]-[G] to get out of the debugger. Take note of what the crash reason is - it should say something like:

   ```
   R1000 system crash at PC=4D42, Error code = 0F
   IOP software crash: IOP_Error

   ...

   Initializing DFS... [OK]
   IO Adapter error - IO Processor failed to check in.
   ```
4. The system will ask if you want a crash dump. Say yes.
5. Run FRU diagnostics on all the boards.
6. If there's still nothing conclusive, give the above information to KEN (also give him the crash dump when available). He may be able to figure out what went wrong.

7. Meanwhile, reboot the system. It may boot normally and stay up for a while, or it may exhibit problems while booting or after booting that gives us better clues as to what the problem is.

Note: In the above example, the problem turned out to be hardware. The message `IO Processor failed to check in` is usually indicative of a hardware problem, and when the machine booted, the serial ports were not accessible. It turned out the IO controller was bad.

♦

TIP: System crashes with IO Processor odd address or bus error	
Applicable to:	*Fix:*
References: CSR 2711	*Keywords:* Address, Boot, Bus, Crash, Dfs, Hang-Up, Iop
Originated from: Rjg	*Revised by:* Vnv

The system crashes during elaboration with:

```
IO Processor Error
IO Processor odd address or bus error
```

This is caused by the IOP software giving the UNIBUS an odd address. The UNIBUS only takes even addresses. The software could have gotten the odd address as a result of;

- Corrupted DFS
- Disk/Tape Controller problem
- Bad cables.
- Bad IOC

Interestingly, if the IOC is bad, the crash may be preceded by other problems:

- The system hangs, where everything - even the break key at the console - has no effect.
- After white-buttoning, you get a message saying:

```
...could not find boot device.
```

♦

HINT: PCM shows PS0 or PS1 lights lit	
Applicable to:	*Fix:*
References:	*Keywords:* Board, Power Supply
Originated from: Vnv	*Revised by:*

On a Model 200 the PCM controls the 5V power supplies PS0 and PS1. When the machine is turned on, the PCM checks the 15V power supply to verify that it is working within range. If it is not then the PCM shuts down the PS0 and PS1. To reset PCM you must cycle power on the machine.

If the PS0 or PS1 light is on, you should check the following items, in this order:

1. Verify that the 15V power supply is outputting +15V. If not replace it.

2. Check that PDU connectors for PS0 and PS1 are delivering 208V (+/-10%). Measure between the two vertical sockets of the plug. If not correct, check power coming to PDU. If this voltage is OK then the PDU is bad.

3. Disconnect the power supply control connector from the backplane for each suspect power supply. Check the output voltages, they should now read +5V. If not, *write protect* the disks, and cycle the power on the machine. If the output is still not +5V then it is likely that the power supply is bad. If If it now reads +5V then the power supply is OK. If the PCM still has PS0 and or PS1 fault lights on then, and if both power supply measured +5V, then PCM is bad, else one or other of the power supplies did not measure +5V, and it should be replaced.

♦

INFO: Series 200 PDU switch	
Applicable to:	*Fix:*
References:	*Keywords:* Pdu, Power Supply
Originated from: Smp	*Revised by:*

There is a 3 position switch - REMOTE, OFF, LOCAL. The PDU has 3 logical sections, Unswitched, Switched 0, Switched 1, which we plug various things into.

Assume: Main breaker is ON.

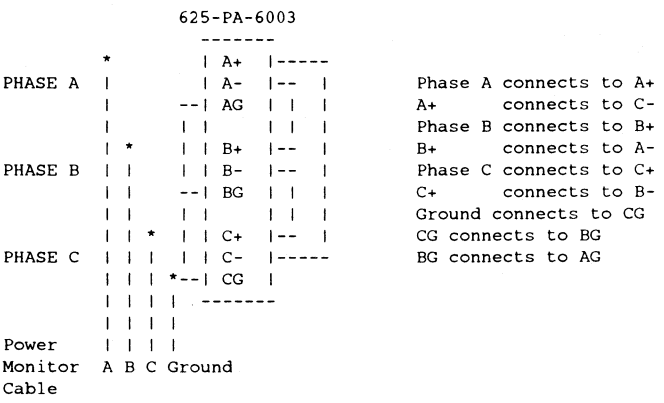
- **REMOTE:** Unswitched plugs are powered (fans, tape, ...). The front panel controls the 2 switched groups. When in DC OFF, only the first group is powered, when in full ON position, both groups are powered.
- **OFF:** All groups (switched and unswitched) are turned OFF. This allows you to control the PDU when servicing the rear.
- **LOCAL:** All groups (switched and unswitched) are turned ON. Opposite of OFF.

◆

INFO: Programming the Dranetz power analyser		
Applicable to:	*Fix:*	
References:	*Keywords:* Pdu, Power Supply	
Originated from: Deg	*Revised by:*	

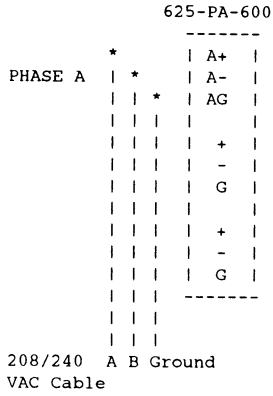
- Monitoring power from the AC Power Monitor Jack

  Hook the power monitor cord to the 626-PA-6003 module according to the diagram below. Connect other end to PDU power monitor jack. Program the Dranetz to monitor either 208/240 VAC. See programming instruction below.



- Monitoring power from the 208/240 VAC Power Jack

  Hook the 208/240 VAC power cord to the 626-PA-6003 module according to the diagram below. Connect other end to a PDU 208/240 VAC jack. Program the Dranetz to monitor either 208/240 VAC. See programming instruction below.



Phase A connects to A+
Phase B connects to A-
Ground connects to AG

- Programming the Dranetz

The Dranetz needs to be programmed to monitor either 208 VAC or 240 VAC depending on the voltage at the site. 240 VAC is rarely used. The voltage high/low limits is +/-10%.

Nominal	Hi	Lo
208VAC	228	188
240VAC	264	216

To program the high/low limits:

1. Turn the function key switch to OPERATE.
2. Press the PROG button.
3. Read printout to determine what channel the 626-PA-6003 module is plugged into.
4. Depress channel # (1-5) that corresponds to step 3 from horizontal row of numbers that starts with SET T.
5. Turn the function key switch to PROGRAM.
6. When the red light comes on, press the HI LIMIT button. Yellow light should light.
7. Input either 228 or 264 on numeric keypad below yellow light. When complete the light next to the ENTER button should light.
8. Press the ENTER button.
9. Repeat steps 6-8 for LO LIMIT button.
10. Turn the printout mode key switch to OUT-OF-LIMITS ONLY.
11. Turn the function key switch to OPERATE.

◆

TIP: How to know if the console is flow controlled		
Applicable to:	*Fix:*	
References:	*Keywords:* Console, Flow Control, Hang-Up, Rs232, Terminal	
Originated from: Jmk, Pam	*Revised by:*	

Lets consider the situation where the machine is running fine but the console is like dead, hitting the break key and asking to redisplay just comes with old stuff. Even swapping the console with another one wouldn't change anything.

In that case the console port might be flow controlled. To determine if this is true, run `Show_Port_Info` at the Kernel. You should get the following kind of display:

```
Kernel: Show_Port_Info
PORT_MANAGER:     INPUT    OUTPUT
                  -----    ------
    BYTES....   3613436 28481996
    PACKETS..    524271  1219043
PORT_ID [1]:
OUTPUT: CLIENT => 70486020; CLIENT IS WAITING; IOP IS BUSY
INPUT: CLIENT => 70502404; FLOW CONTROL ENABLED
```

The line

```
OUTPUT: CLIENT => 70486020; CLIENT IS WAITING; IOP IS BUSY
```

indicates that XOFF has been sent. Some terminals will get confused and forget that they did XOFF, and then not send XON even if you type [Ctrl][Q] (the terminal doesn't actually send anything when you type this key combination if it thinks it's unneccessary).

The thing to try is to run:

```
Terminal.Set_Flow_Control (1, "NONE");
Terminal.Set_Flow_Control (1, "XON_XOFF");
```

to un-control the console port.

♦

INFO: Programming the Facit terminal	
Applicable to:	*Fix:*
References:	*Keywords:* Facit, Terminal
Originated from: Pam	*Revised by:*

1. Terminal setup

   - Enter setup mode B: press [Set up] and then [5].
   - Move the cursor to group 2.
   - Change the settings for the third switch, `BACKSPACE KEY`: press [6].
   - Move the cursor to group 4.
   - Change switches `PROTOCOL`, `XON/XOFF FROM HOST`, and `PRINTER => HOST`: press [6] for each.
   - Toggle to setup mode C: press [5].
   - Move the cursor to group 2.
   - Change the last switch in this group, `BIG/FLIP PAGE`: press [6].
   - Save the settings: press [Shift][S].

   The screen displays a waiting message for 3 seconds. After the mode settings are saved, quit setup mode by pressing [Set up].

2. Programming the keyboard.

   To store the Rational custom key bindings, EAROM storage space must be freed from other factory default key codes. The following procedures describe how to allocate this memory on terminals that have not been programmed by Rational.

   - Allocating terminal memory

     Programming the 12 function keys require 108 bytes of storage space. Deleting the factory default key mappings from the custom setup modes 1 and 2 frees 33 bytes of EAROM, making 108 bytes available. The following step-by-step procedure clears the required memory:

     – Enter setup mode A: press [Set up].

     – Enter custom setup mode 1: press [Ctrl][Set up].
     Note the `EAROM BYTES LEFT`: display in the upper-right corner of the screen. It should list 75 bytes. Before Rational function keys can be programmed, at least 108 bytes must be displayed.The `CUSTOMIZED CONTROL CODES` in the center of the screen, indicating that 14 control codes are currently stored in setup mode 1.

- Delete one customized control code: press [Ctrl][Del].
  Note that the EAROM BYTES LEFT: display has increased by 2. The CUSTOMIZED CONTROL CODES display has decreased by 1.

- Repeat this procedure 13 times: press [Ctrl][Del].
  This exhausts all control code codes in custom mode 1 and increases the available EAROM bytes to 103. The 5 bytes still required can be opened by following the same procedure in setup mode 2.

- Enter custom mode setup 2: press [5].

- Delete one customized escape code: press [Ctrl][Del].

- Repeat this procedure twice.
  Note that 109 bytes of EAROM show in the EAROM BYTES LEFT display. This is sufficient memory to program the Rational function keys.

- Leave the custom setup mode: press [Set up].
  The terminal enters standard setup mode A.

• Programming the function keys

The following steps describe how to program the 12 function keys.

- Enter setup mode B: press [5].

- Program each of the function keys by pressing [Ctrl][F*n*] (where *n* ranges from 1 through 12), followed by the three-character sequence for the function key from the following table, and [Ctrl][CR].

Function key	Programming sequence
F1	Esc OE
F2	Esc OF
F3	Esc OG
F4	Esc OH
F5	Esc OI
F6	Esc OJ
F7	Esc OK
F8	Esc OL
F9	Esc ON
F10	Esc OO
F11	Esc OT
F12	Esc OU

- After all keys have ben programmed, save them: press [Shift][S].

◆

INFO: How to connect a MVME135 to a R1000 via a DECserver

Applicable to:	*Fix:*
References:	*Keywords:* Cable, Connector, Decnet, Mc68K, Rs232, Terminal Server
Originated from: Hts, Rjg	*Revised by:*

DECserver port settings on both the target box and R1000 side:

```
Character Size:            8          Input Speed:        9600
Flow Control:            CTS          Output Speed:       9600
Parity:                 None          Modem Control:   Disabled

Access:                Remote          Local Switch:       None
Backwards Switch:        None          Name:             PORT_1
Break:                 Remote          Session Limit:         4
Forwards Switch:         None          Type:               Soft

Preferred Service: None

Authorized Groups:   0
(Current)  Groups:   0

Enabled Characteristics:

Autoconnect,  Input Flow Control,  Output Flow Control
```

The above assumes that you will form the connection by issuing a connect command from some local port on a DECserver; else there may be differences in the settings for `Break` and `Access`.

Pinouts for cable connectors:

```
MVME135  Port  DEC Server Port   Enet  DEC Server Port     R1000 Port
                                   \
  Signal  Pin       Pin  Signal    /   Signal  Pin       Pin  Signal
                                   \
    CTS    8---------4    RTS      /    RTS     4---------5    CTS
    RxD    2---------2    TxD      \    RxD     3---------2    TxD
    TxD    3---------3    RxD      /    TxD     2---------3    RxD
    RTS    7---------5    CTS      \    CTS     5---------4    RTS
    Gnd    5---------7    Gnd      /    Gnd     7---------7    Gnd
    DCD    1---------20   DTR      \    DSR     6--+
    DTR    4------+--8    DCD      /    DCD     8--+------20   DTR
                  +--6    DSR      \    DTR     20--------8    DCD
```

♦

TIP: How to connect a LaserWriter to a R1000	
Applicable to:	*Fix:*
References:	*Keywords:* Cable, Connector, Laser, Postscript, Printing
Originated from: Bed	*Revised by:*

This is how to set up an **Apple LaserWriter Plus** for PostScript operation when direct-connected to the R1000 (assuming port 31 is used on the R1000).

Set the LaserWriter rotary/baud-rate switch (located by the serial connector) to 9600 (baud, postscript). The *special* setting is an ASCII-based printer emulation that can be used prior to setting up the spooler for PostScript printing.

At the R1000, use the following wiring for the DB-25 connector (internally known as a D-type connector):

Pin Number	4	7	3	2	8	20	6	5
RS-232/C Signal	RTS	GND	RXD	TXD	DCD	DTR	DSR	CTS
Wire Color Code (Internal only)	SLATE	BROWN	YELLOW	GREEN	RED	BLACK	ORANGE	BLUE

At the LaserWriter, use the following wiring for the DB-25 connector (internally known as an AM-type connector):

Pin Number	8	7	2	3	4	5	6	20
RS-232/C Signal	DCD	GND	TXD	RXD	RTS	CTS	DSR	DTR
Wire Color Code (Internal only)	SLATE	BROWN	YELLOW	GREEN	RED	BLACK	ORANGE	BLUE

Using operations from package `Terminal` set the port as follows:

```
Terminal Settings for Port 31
-----------------------------
Terminal Type =                    RATIONAL
Input Baud Rate =                  BAUD_9600
Output Baud Rate =                 BAUD_9600
Parity =                           NONE
Stop_Bits =                         1          *** Note!
Char_Size =                        CHAR_8
Flow Control for Transmit Data =   XON_XOFF
Xon =                              DC1
Xoff =                             DC3
Flow Control for Receive Data =    NONE
Login_Disabled =                   TRUE
Log_Failed_Logins =                FALSE
Disconnect_On_Failed_Login =       FALSE
Disconnect_On_Logoff =             FALSE
```

```
Logoff_On_Disconnect =          FALSE
Disconnect_On_Disconnect =      FALSE
```

Add the device:

```
Q.Add (Device => "Terminal_31", Options => "XON_XOFF, Laser_Comm");
```

Create a printer class:

```
Q.Create (Class => "LaserWriter_Plus");
```

Register the printer class:

```
Q.Register (Device => "Terminal_31", Class => "LaserWriter_Plus");
```

Optionally, set it as the default:

```
Q.Default (Class => "LaserWriter_Plus");
```

Enable the port/device:

```
Q.Enable (Device => "Terminal_31");
```

Double-check the results:

```
Q.Devices (Which => "all", Show_State => True, Show_Classes => True);
```

...should produce a report containing:

```
   Device      Protocol  Characteristics   State       Classes
 ===========  ========  ===============  =======  ================
 TERMINAL_31  XON_XOFF     Laser_Comm     Enabled  LASERWRITER_PLUS
```

```
Q.Classes (Which => "all", Show_Devices => True);
```

...should produce a report containing:

```
      Class          Device(s)
 ================  ===========
 LASERWRITER_PLUS  TERMINAL_31
```

The following is an example of the `Queue.Print` command when used to print using the PostScript mode (for more detail, see the documentation for the `Queue.Print` command and/or the comments at the end of package `!Commands.Queue`):

```
Q.Print (Name => "<IMAGE>",
      Options =>
         "POSTSCRIPT=>(FORMAT=>Automatic, TWOUP=>True, BORDER=>True, " &
         "FILENAME=>True, DATE=>True, PAGES=>1..Integer'Last, " &
         "REVERSED=>True), COPIES=>1, CLASS=>LaserWriter_Plus",
      Banner => "<DEFAULT>",
      Header => "<DEFAULT>",
      Footer => "<DEFAULT>");
```

♦

INFO: Printer switch information	
Applicable to:	*Fix:*
References:	*Keywords:* Connector, Printer Switch
Originated from: Cjj, Deg	*Revised by:*

The purpose of the printer switch is to allow two R1000 CPUs to share a single printer. It is standard on the Series 200 Model 40. The printer switch includes a hardware component (the print switch) plus special software for both CPUs.

1. **Theory of operation**

   When a CPU wants to use the printer, it signals the printer switch by turning the Request-To-Send (RTS), Pin 4, ON at the R1000 communications panel.

   The printer switch responds by routing the Clear-To-Send (CTS), Pin 5, from the printer to the communications panel. When the CTS goes ON, the CPU transmits a print job. The CPU signals the end of the print job by turning the RTS OFF for a minimum of 3 seconds.

   The printer switch hardware arbitrates between the two CPUs, and grants the printer to only one at a time. If the printer has been granted to one CPU (the owner), then the other CPU (the non-owner) may request the printer by turning RTS ON, but will see CTS OFF until the owner turns RTS OFF, at which time the printer will be granted to the new owner.

   When the printer becomes ready and the buffer is near empty, the printer raises Data-Terminal-Ready (DTR) signaling the host to start transmitting data. If XON_XOFF protocol is specified then the printer transmits an XON character when DTR is raised. When the printer is not ready or the buffer becomes full, the printer drops DTR signaling the host to stop transmitting data. If XON_XOFF protocol is specified then the printer transmits an XOFF character 0.5 seconds before DTR is dropped.

   Note: DTR on the printer is tied into CTS. See printer cable diagram below.

2. **Installation**

   A Rational-supplied ribbon cable must be connected from each CPU communications panel to the printer switch. These cables provide straight-through connections for pins 1, 2, 3, 4, 5, and 7. Each cable connects the printer port (usually port 31, the right-most port on the first communication panel) to its corresponding printer switch port (female RS-232 connector). Connect the output port of the printer switch to the printer using the standard Rational printer cable. The standard Rational printer cable is configured as follows:

```
Printer (DTE)          Switch or Comm Panel (DTE)

GND 1  ----------------  1

 TD 2  >---------------->  3 RD

 RD 3  <----------------<  2 TD
```

```
GND 7  ------------------  7

CTS 5  <----------------->  5
         |
DCD 8  <--              -->  8
         |              |
DTR 20 >--              --<  20
```

The output pin configuration of the printer switch is as follows:

```
1   -- GND

2   -- TX data

3   -- RX data

7   -- GND

5   -- CTS
```

No other modem control signals are provided from the printer switch output.

3. **Configuration**

The printer and communication panel must be configured for hardware flow control.

- Printer

  The printer should be setup for DTR protocol. This is the default setting. XON_XOFF protocol will work if the port flow control for transmit data is XON_XOFF which it is by default when the spooler device is enabled with RTS protocol.

- Communication panel

  DIP Switch SW-1 position 5 must be ON (to the right). This causes a wait for CTS before transmitting data.

- Print spooler device

  Configured for RTS protocol:

```
Queue.Create("LP");
Queue.Add(Device => "Terminal_31", Options => "RTS");
Queue.Register(Device => "Terminal_31", Class => "LP");
```

INFO: Printer switch information	
Applicable to:	*Fix:*
References:	*Keywords:* Connector, Printer Switch
Originated from: Cjj, Deg	*Revised by:*

The purpose of the printer switch is to allow two R1000 CPUs to share a single printer. It is standard on the Series 200 Model 40. The printer switch includes a hardware component (the print switch) plus special software for both CPUs.

1. **Theory of operation**

   When a CPU wants to use the printer, it signals the printer switch by turning the Request-To-Send (RTS), Pin 4, ON at the R1000 communications panel.

   The printer switch responds by routing the Clear-To-Send (CTS), Pin 5, from the printer to the communications panel. When the CTS goes ON, the CPU transmits a print job. The CPU signals the end of the print job by turning the RTS OFF for a minimum of 3 seconds.

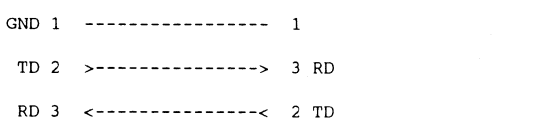
   The printer switch hardware arbitrates between the two CPUs, and grants the printer to only one at a time. If the printer has been granted to one CPU (the owner), then the other CPU (the non-owner) may request the printer by turning RTS ON, but will see CTS OFF until the owner turns RTS OFF, at which time the printer will be granted to the new owner.

   When the printer becomes ready and the buffer is near empty, the printer raises Data-Terminal-Ready (DTR) signaling the host to start transmitting data. If XON_XOFF protocol is specified then the printer transmits an XON character when DTR is raised. When the printer is not ready or the buffer becomes full, the printer drops DTR signaling the host to stop transmitting data. If XON_XOFF protocol is specified then the printer transmits an XOFF character 0.5 seconds before DTR is dropped.

   Note: DTR on the printer is tied into CTS. See printer cable diagram below.

2. **Installation**

   A Rational-supplied ribbon cable must be connected from each CPU communications panel to the printer switch. These cables provide straight-through connections for pins 1, 2, 3, 4, 5, and 7. Each cable connects the printer port (usually port 31, the right-most port on the first communication panel) to its corresponding printer switch port (female RS-232 connector). Connect the output port of the printer switch to the printer using the standard Rational printer cable. The standard Rational printer cable is configured as follows:



```
GND 7  ------------------  7

CTS 5  <----------------->  5
         |
DCD 8  <--              -->  8
         |              |
DTR 20 >--              --<  20
```

The output pin configuration of the printer switch is as follows:

```
1  -- GND

2  -- TX data

3  -- RX data

7  -- GND

5  -- CTS
```

No other modem control signals are provided from the printer switch output.

3. **Configuration**

The printer and communication panel must be configured for hardware flow control.

- Printer

  The printer should be setup for DTR protocol. This is the default setting. XON_XOFF protocol will work if the port flow control for transmit data is XON_XOFF which it is by default when the spooler device is enabled with RTS protocol.

- Communication panel

  DIP Switch SW-1 position 5 must be ON (to the right). This causes a wait for CTS before transmitting data.

- Print spooler device

  Configured for RTS protocol:

```
Queue.Create("LP");
Queue.Add(Device => "Terminal_31", Options => "RTS");
Queue.Register(Device => "Terminal_31", Class => "LP");
Queue.Enable("Terminal_31");
Queue.Default("LP");
```

Note: If either of the CPUs sharing the printer is being booted, the other CPU cannot print. This is because, during the boot process the CPU turns RTS ON on all ports. The print Switch grants the printer to the booting CPU, even though it is not printing.

Note: If either of the CPUs sharing the printer has not enabled (using `Queue.Enable`) the spooler device to which the printer switch is connected, the other CPU cannot print. This is because the RTS signal is not turned OFF until the device is enabled.

♦

INFO: How to configure the Series 100/200 communication panel	
Applicable to:	*Fix:*
References:	*Keywords:* Communication Panel, Rs232
Originated from: Deg, Rfg	*Revised by:*

- The 4 pole dip switch (SW3) located nearest the front of the communication controller board needs to be set for the number of communication panels (if it is not properly set, the corresponding ports will appear to be dead).

  For one communication panel, SW3 needs to be set as follows: 1,2,3 Open, 4 Closed. For two communication panels, SW3 needs to be set as follows: 1,3 Open, 2, 4 Closed. *Closed* means the position nearest to the numbers on the dip switch.

- The communication panel can be configured to use hardware flow control by setting its switch 1 (on the left of the front panel), bit 5 to on, and rebooting the machine. *On* is on the right side of the communication panel. The communication panel will then stop sending data while the CTS line is low.

Warning: If you have other devices hooked up to the communication panel, such as terminals, you need to be sure that the connectors are wired properly. For example, you could set up the terminal for hardware flow control, making sure that DTR from the terminal goes to CTS on the communication panel. Another option is to have a loopback from RTS to CTS on the communication panel connector. Devices using this type of connector would disable hardware flow control - since RTS is always high, the communication panel would always send data.

If the other devices hooked up to the communication panel are not essential, you may want to simply flip the switch and reboot. There is a good chance that the connectors are set up properly, and everything may just work immediately. On the other hand, if some of these devices are essential, you may want to check the connectors and carefully consider the effects of introducing hardware flow control. At this point, it is unfortunately not possible to set up only one port for hardware flow control.

♦

INFO: How to reboot an Ethernet controller		
Applicable to:	*Fix:*	
References:	*Keywords:* Board, Boot, Ethernet, Network	
Originated from: Jmk, Vnv	*Revised by:*	

Here's the safe way to reboot an Ethernet controller.

1. Disconnect the machine from its Ethernet transceiver. A convenient way to do this is to unplug the transceiver cable from the connector on the R1000 external bulkhead.
2. Login to an RS-232 port, or to the console `CI`, as `Operator`.
3. Run `Network.Close_All`.
4. Run `Network.Show`, which will produce a listing of connections owned by server jobs. One column in the listing is the job number: `Job.Kill` each listed job, excepting jobs 4 and 5.
5. Run `Network.Close_All` again.
6. Take a snapshot `Daemon.Run ("Snapshot")`; the next step has a small chance of crashing the machine.
7. Run `!Machine.Initialize_Network`.
8. Reconnect the machine to its Ethernet transceiver.

◆

INFO: Connector pinouts

Applicable to:	*Fix:*
References:	*Keywords:* Cable, Connector, Rs232
Originated from: Bobr, Deg	*Revised by:*

	SLATE	BROWN	YELLOW	GREEN	RED	BLACK	ORANGE	BLUE
KEY	8	7	2	3	4	5	6	20
	DCD	GND	TXD	RXD	RTS	CTS	DSR	DTR

male - please label these "AM"

	SLATE	BROWN	YELLOW	GREEN	RED	BLACK	ORANGE	BLUE
KEY	8	7	2	3	4	5	6	20
	DCD	GND	TXD	RXD	RTS	CTS	DSR	DTR

female - please label these "AF"

	SLATE	BROWN	YELLOW	GREEN	RED	BLACK	ORANGE	BLUE
KEY	-	7	2	3	20	5	-	-
		GND	TXD	RXD	DTR	CTS		

jumper slate, orange, and blue wires and wrap in tape

female - please label these "BF"

	SLATE	BROWN	YELLOW	GREEN	RED	BLACK	ORANGE	BLUE
KEY	-	7	3	2	5	4	20	6
		GND	RXD	TXD	CTS	RTS	DTR	DSR

crimp pin to the slate wire but wrap in tape

male - please label these "C"

	SLATE	BROWN	YELLOW	GREEN	RED	BLACK	ORANGE	BLUE
KEY	4	7	3	2	8	20	6	5
	RTS	GND	RXD	TXD	DCD	DTR	DSR	CTS

female - please label these "D"

All connectors are used with straight through cables.

Use "A" and "D" connectors for DTE to DTE connections
Use "A" and "C" connectors for DCE to DCE connections
Use "A" and "B" connectors for DCE to DTE connections with Hardware Flow Control
Use "A" and "A" connectors for DCE to DTE connections

DTE Equipment:

Rational Line Printer, Communication Panel,Terminals

DCE Equipment:

Bridge Box, Modems

◆

TIP: How to connect a printer on a Facit terminal

Applicable to:	*Fix:*
References:	*Keywords:* Cable, Connector, Facit, Printing, Terminal
Originated from: Hts	*Revised by:*

The following connections should be made between the Facit printer port and the printer:

```
FACIT PRINTER PORT                                          PRINTER
------------------                                          -------

RXD (pin 2) <-------------------------------------------> TXD  (pin 2)

TXD (pin 3) <-------------------------------------------> RXD  (pin 3)

GND (pin 7) <-------------------------------------------> GND  (pin 7)

CTS (pin 5) <---                                    ---> CTS  (pin 5)
                |                                  |
READY/BUSY  <---                                    ---> DSR  (pin 6)
  (pin 19)                                         |
(See note below)                                    ---> RLSD (pin 8)
                                                   |
                                                    ---> DTR  (pin 20)
```

Note: The standard *AF D*-type connectors do not have a connection to pin 19, so it may be necessary to use a *silver* converter which links pins 19 and 20 and then make the link between pin 5 and pin 20 in the *AF* connector.

The following set-up switch configuration is required at the Facit:

SET-UP B

```
B:3.2 = 1   Enables status line to allow printer status to be shown.
B:4.4 = 1   All data received from the printer, except XOFF/XON ignored.
```

SET-UP C

```
C:4.1 = 1   Enables display of printer status on status line.
C:5.1 = 0   Extent of print operation determined by Print Extent Switch
            (C:5.2).
C:5.2 = 0   Print scrolling region.
C:5.3 = 1   Print all data regardless of cursor position.
C:5.4 = 0   1 stop bit.
C:6.1 = 1   8 data bits.
C:6.2 = 0   8th data bit = 0.
C:6.3 = X   Don't care.
C:6.4 = 0   Parity disabled.
```

Printer Baud rate = 9600 baud.

CUSTOM MODE SET-UP 4 (Use [Ctrl] [Set up] + 3 x [Set up A/B/C] from Set-up A)

```
CM:2.1 = 1  Print terminated by a formfeed.
CM:2.2 = 0  Screen control codes not sent.
CM:2.4 = 1 [Shift] [Enter] prints a page.
CM:3.1 = 0  Print active page.
```

This configuration can be saved from Set-up A by typing [Shift] [S].

It is also important that the printer is set up to use the XOFF/XON protocol (ie. PRTCOL = XON/OF).

To dump the contents of the screen to the printer type [Shift] [Enter], with the cusor positioned anywhere on the screen. The printer status will show `part` in reverse video whilst the operation is taking place. It may sometimes be necessary to refresh the screen ([Ctrl] [L]) before printing to cause the entire screen to be sent.

Auto-print mode can be selected and deselected by typing [Ctrl] [Enter]. This causes each line of data on the screen to be printed whenever the cursor is moved off the line (ie. by LF,FF or VT characters). This mode is particularly useful for providing a hardcopy of the console output. The printer status will display `auto`.

♦

INFO: RS232 signals and the R1000	
Applicable to:	*Fix:*
References:	*Keywords:* Cable, Connector, Rs232
Originated from: Deg	*Revised by:*

The R1000 Communications Panel is a DTE device that understands only a subset of the RS-232 interface.

GND 1 - Protective Ground: Cable and connector shields.

TXD 2 (out) - Transmitted Data: Transmitted Data maybe either Hardware or Software flow controlled.

RXD 3 (in) - Received Data: Received Data may only be Software flow controlled. Hardware flow control for RXD is not supported by the R1000.

RTS 4 (out) - Request To Send: Controlled by port manager if RTS protocol is specified else RTS is always ON.

CTS 5 (in) - Clear To Send: If R1000 is configured for hardware flow control, CTS must be ON before the R1000 will transmitted data. If CTS is turned off, the R1000 will stop transmitting data by the next character.

DSR 6 (in) - Data Set Ready: Not used by the R1000. Any other signal not mentioned here is also not used by the R1000.

GND 7 - Signal Ground: Other signal grounds are measured relative to the ground.

DCD 8 (in) - Data Carrier Detect: If this signal is toggled from ON to OFF, the R1000 interprets this as an incoming disconnect event. See Disconnect_On_Disconnect and Logoff_On_Disconnect.

DTR 20 (out) - Data Terminal Ready: After the R1000 has been initialized, this signal is turned ON. The R1000 may request a disconnect event by switching this signal OFF for 3 seconds and then back ON. See Disconnect_On_Disconnect, Disconnect_On_Logoff and Disconnect_On_Failed_Login.

◆

INFO: R1000 300/S thermal sensor wiring		
Applicable to:	*Fix:*	
References:	*Keywords:* Cable, Sensor, Wiring	
Originated from: Alm, Gbd, Rws, Wrj		*Revised by:*

The thermal sensors are temperature-sensitive switches that are closed at temperatures up to a specified threshold temperature. Above the threshold temperature, the switches are open. There are two types of sensors. One type has red dots on top of each sensor and opens the circuit at 50°C (122°F). The other type has green dots and opens at 55°C (131°F).

Each R1000 300S has six sensors, three of each type. The sensors are mounted in three pairs. Each pair has one sensor of each type. The sensors are mounted at these locations:

- Top of the CPU enclosure ("CPU bay")
- top of the disk enclosure ("peripheral bay")
- other side of the disk enclosure

The three sensors of each type are wired in an independent circuit from the other type. The sensors are wired in series so that any one sensor can open its circuit.

The wires are sheathed in a four-conductor cable between the CPU enclosure and the disk enclosure. (beware: manufacturing has not been consistent in color-coding the wires). There is another cable between the sensor assembly in the CPU enclosure and the backplane connector where the wires terminate.

The backplane connector is a nine-position connector in which four pins are actually used. The following diagram shows the backplane connector (the connector that is mounted on the backplane as viewed from behind the R1000 with the cable unplugged):

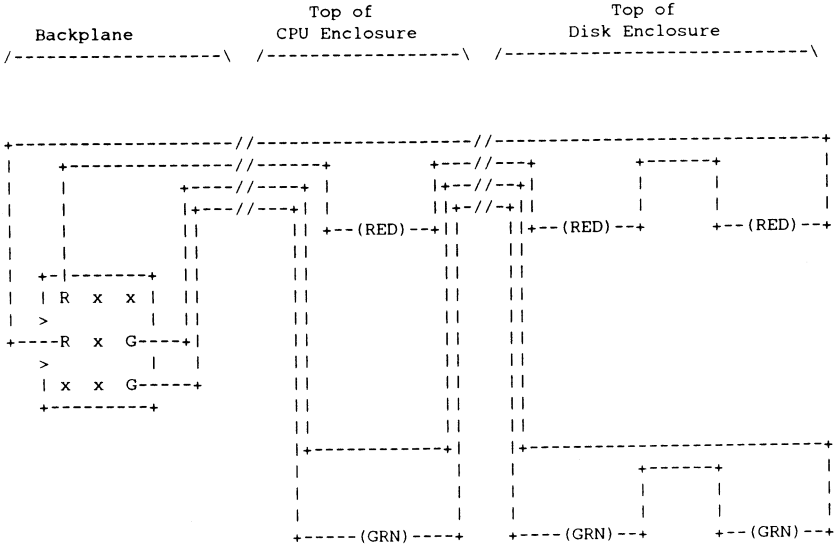
```
          top
       +----------+
       | o  x  x  |
       >          |
left   | o  x  o  | right (as viewed from rear of R1000)
       >          |
       | x  x  o  |
       +----------+
```

The > characters indicate the location of the *keys*, shallow channels in the plastic connector shells that make one side of the square connectors unique to ensure that the proper pins are mated when the cable is plugged in.

The x characters indicate unused pin positions (as determined by inspection of the cable connector).

The o characters indicate pin positions that are used.

The diagram below shows how the thermal sensor circuits are wired.



Symbol	Meaning
//	sheathed control cable
(RED)	50°C thermal sensor (red dot on top)
(GRN)	55°C thermal sensor (green dot on top)
x	unused pin position
R	pin for red sensor circuit
G	pin for green sensor circuit

Note that although the picture shows a particular polarity for the two R (red circuit) pins, the polarity is unimportant. What matters is that the red circuit be wired to the two pins closest to the key channels on the connector. Similarly, the two G (green circuit) pins might be reversed.

♦

TIP: Configuring a Concord Data Systems 224 modem		
Applicable to:	*Fix:*	
References:	*Keywords:* Modem	
Originated from: Deg, Tab, Tdg	*Revised by:*	

- Connection to a terminal

  Connect the CDS modem to a terminal using a straight through cable. Set up terminal for 2400 baud, 8 data bits, no parity.

- Connection to a R1000

  Connect the CDS modem to a Communications Panel port using a straight through cable. Configure the port as follows:

```
Terminal Type =                    RATIONAL
Input Baud Rate =                  BAUD_2400
Output Baud Rate =                 BAUD_2400
Parity =                           NONE
Stop_Bits =                         2
Char_Size =                        CHAR_8
Flow Control for Transmit Data =  XON_XOFF
Flow Control for Receive Data =   NONE
```

- CDS Series I Autodial

  If you need to connect a CDS Series I Autodial modem to a R1000 so that it can receive incoming calls, you have to set the defaults so that you cannot dial out from the R1000 over that modem: it will only answer calls.

  The reason is that the R1000 sends out a `commence login` message that the modem interprets as a command string from the local host. Once it starts to receive a command string, it will not answer incoming calls. The modem must be set to *dial-in only* mode so that it will not recognize command strings from the local host. To do this you set switch 4-8 to `on`. This switch tells the modem to ignore command strings from the local host.

- CDS Series II

  Type `AT` to get the attention of the modem. Type `?` to list current settings. The following is a list of settings needed to communicate with an R1000. Stars (`*`) specify values that are different from the factory settings.

```
CONCORD DATA SYSTEMS 224 RM16 -  REV 2.05
   * MON :0          * SPD :5     -- 8 bits, no parity
     BLK :256          SPE :0
     RLB :1          * SPF :2     -- Flow Control
     SBA :0            SRD :1
   * SBK :5            SRG :001
     SCC :0            SRI :0
     SCM :0          * SRM :2     -- Echo off
```

```
  SCP :1          SSD :1
* SE  :043        SST :0
  SFP :0
* SLCA:2          STA :005
* SLCB:2          STB :007
* SLCC:2          STC :030
* SLCD:1          STD :000
* SLCF:2          STE :025
  SLT :1
  SMB :2400     * STI :060
  SMD :2          STP :002
* SMF :0          STR :000
  SMT :3          STW :002
* SPB :2400       STX :0
```

SRM 2 will set the CDS modem so that commands and responses are not echoed. After setting the above command values type SAVE to store values in nonvolatile memory.

INFO: PAD Parameters

Applicable to:	*Fix:*
References: Memotec SP/830 PAD - System Reference Manual	*Keywords:* Irae, Pad, X25
Originated from: Memotec	*Revised by:* Mv

1	PAD RECALL USING A CHARACTER	0	Escape from data transfer mode not permitted.
		1	Escape from data transfer mode permitted when PAD receives DLE character.
		2 to 255	Escape from data transfer mode permitted when PAD receives graphic character defined by the user.
2	ECHO	0	Echo function disabled.
		1	Echo function enabled.
3	SELECTION OF DATA FORWARDING SIGNALS	0	No data forwarding signal recognized.
		2	PAD forwards data packet each time a CR is received form the user.
		1, 3 to 125, 127 to 255	See parameter EN of SCL's Set Character Definition command.
		126	PAD forwards data packet each time a control character (00 to 1F) or DEL (delete) character is received from the user.
4	SELECTION OF IDLE TIMER DELAY	0	Idle timer disabled.
		1 to 255	Value of idle timer delay in twentieths of a second.
5	ANCILLARY DEVICE CONTROL	0	Flow control between PAD and user's device disabled.
		1	Flow control between PAD and user's device enabled.
6	CONTROL OF PAD SERVICE SIGNALS	0	No PAD service signals sent to user.
		1	Only PAD service signals sent to user.
		4	Only Prompt PAD service signal sent to user.
7	OPERATION OF PAD ON RECEIPT OF BREAK SIGNAL	0	No response from PAD on receipt of Break signal.
		1	PAD sends interrupt packet to remote end.
		2	PAD resets virtual call.
		4	PAD send indication of Break

			message to remote end.
		8	PAD responds by escaping from data transfer mode and entering command mode.
		16	PAD discards all data received from user's device.
8	DISCARD OUTPUT	0	PAD sends data to user's device.
		1	PAD discards data pending for user's device.
9	PADDING AFTER CR	0	PAD does not insert padding characters.
		1 to 255	PAD inserts the specified number of padding characters.
10	LINE FOLDING	0	PAD does not insert CR and LF.
		1 to 255	PAD inserts CR and LF after specified number of graphic characters.
11	SPEED OF USER PORT	0	110 bits/second.
		1	134.5 bits/second.
		2	300 bits/second.
		3	1200 bits/second.
		4	600 bits/second.
		5	75 bits/second.
		6	150 bits/second.
		12	2400 bits/second.
		13	4800 bits/second.
		14	9600 bits/second.
12	FLOW CONTROL OF PAD BY USER'S DEVICE	0	User's device does not transmit PAD flow control characters to PAD.
		1	User's device transmits flow control characters to PAD.
13	LINEFEED INSERTION AFTER CR	0	No linefeed (LF) insertion.
		1	PAD inserts LF after transmission of CR in data stream received from

INFO: PAD Parameters

Applicable to:	*Fix:*
References: Memotec SP/830 PAD - System Reference Manual	*Keywords:* Irae, Pad, X25
Originated from: Memotec	*Revised by:* Mv

1	PAD RECALL USING A CHARACTER	0	Escape from data transfer mode not permitted.
		1	Escape from data transfer mode permitted when PAD receives DLE character.
		2 to 255	Escape from data transfer mode permitted when PAD receives graphic character defined by the user.
2	ECHO	0	Echo function disabled.
		1	Echo function enabled.
3	SELECTION OF DATA FORWARDING SIGNALS	0	No data forwarding signal recognized.
		2	PAD forwards data packet each time a CR is received form the user.
		1, 3 to 125, 127 to 255	See parameter EN of SCL's Set Character Definition command.
		126	PAD forwards data packet each time a control character (00 to 1F) or DEL (delete) character is received from the user.
4	SELECTION OF IDLE TIMER DELAY	0	Idle timer disabled.
		1 to 255	Value of idle timer delay in twentieths of a second.
5	ANCILLARY DEVICE CONTROL	0	Flow control between PAD and user's device disabled.
		1	Flow control between PAD and user's device enabled.
6	CONTROL OF PAD SERVICE SIGNALS	0	No PAD service signals sent to user.
		1	Only PAD service signals sent to user.
		4	Only Prompt PAD service signal sent to user.
7	OPERATION OF PAD ON RECEIPT OF BREAK SIGNAL	0	No response from PAD on receipt of Break signal.
		1	PAD sends interrupt packet to remote end.
		2	PAD resets virtual call.
		4	PAD send indication of Break

			message to remote end.
		8	PAD responds by escaping from data transfer mode and entering command mode.
		16	PAD discards all data received from user's device.
8	DISCARD OUTPUT	0	PAD sends data to user's device.
		1	PAD discards data pending for user's device.
9	PADDING AFTER CR	0	PAD does not insert padding characters.
		1 to 255	PAD inserts the specified number of padding characters.
10	LINE FOLDING	0	PAD does not insert CR and LF.
		1 to 255	PAD inserts CR and LF after specified number of graphic characters.
11	SPEED OF USER PORT	0	110 bits/second.
		1	134.5 bits/second.
		2	300 bits/second.
		3	1200 bits/second.
		4	600 bits/second.
		5	75 bits/second.
		6	150 bits/second.
		12	2400 bits/second.
		13	4800 bits/second.
		14	9600 bits/second.
12	FLOW CONTROL OF PAD BY USER'S DEVICE	0	User's device does not transmit PAD flow control characters to PAD.
		1	User's device transmits flow control characters to PAD.
13	LINEFEED INSERTION AFTER CR	0	No linefeed (LF) insertion.
		1	PAD inserts LF after transmission of CR in data stream received from network to user port.
		2	PAD inserts LF in data stream to be transmitted over the network after receiving a CR from user port.
		4	PAD transmits LF to the user port after echoing a CR.
14	PADDING AFTER LF	0	PAD does not insert padding characters after LF.
		1 to 255	PAD inserts number of specified padding characters after LF.
15	EDITING	0	No editing during data transfer mode.
		1	Editing permitted during data transfer mode.

16	CHARACTER DELETE	0 to 255	One IA5 character representing the character-delete character.
17	LINE DELETE	0 to 255	One IA5 character representing the line-delete character.
18	LINE DISPLAY	0 to 255	One IA5 character representing the line-display character.
19	SELECTION OF PAD SERVICE SIGNALS FOR EDITING	1	PAD service signals for DTEs with printing representation (xxx).
		2	PAD service signals for video display DTE's (BS, SP, BS).
20	ECHO MASK	1	Alphanumeric characters.
		2	CR.
		4	ESC, BEL, ENQ, ACK.
		8	DEL, CAN, DC2.
		16	ETX, EOT.
		32	HT, LF, VT, FF.
		64	All other characters in columns 0 and 1 not included above.
		128	All other characters not included above.
		256	All characters.

◆

INFO: Initioa - international version	
Applicable to:	*Fix:*
References:	*Keywords:* Cli, Ioa, Irae, Pad, X25
Originated from: Ken	*Revised by:*

```
CLI> x initioa
Enter CLUSTER ID [902162] :
```

Diagnostic connection may be made either via the internal modem or X.25 PAD.

- For internal modem specify `M` as the connection type
- For X.25 PAD specify `X`

```
Enter connection type [X] :
Enter remote diagnostic X.25 DNIC [310690437600] :
Enter TYMNET password [oscar@anyone] :
Enter AUTODIAL command string for far end connection to RATIONAL.
Enter AUTODIAL command [8008416400%%] :
CLI>
```

♦

INFO: Creating a slew tape	
Applicable to: All releases	*Fix:*
References:	*Keywords:* Cli, Disk, Slew, Tape
Originated from: Slr	*Revised by:* Mv

A *slew* tape is a tape used to configure the disk controller for a given type of disk drive.
To create a slew tape for the *Series 100*:

```
CLI> DUMP TAPE_BOOSTRAP.M100, KERNEL_0.M100, SLEW.M100
```

and for the *Series 200*:

```
CLI> DUMP DFS_BOOTSTRAP.M200, KERNEL_0.M200, FS_0.M200, SLEW.M200
```

Note: this is different from a *skew* tape, used to tune a tape drive and that can only be bought.
♦

INFO: Microcode trace for bus parity error problems	
Applicable to:	*Fix:*
References:	*Keywords:* Board, Cli, Crash, Microcode, Parity Error
Originated from: Mam	*Revised by:*

The R1000 crashes and displays a *postmortem* message similar to the following:

```
Val has detected a(n) a bus parity error
FUI has detected a(n) TYP bus parity error
MEM1 has detected a tag store parity error
```

Running the FRUs fails to find a problem.

The solution is to get a microcode trace before you do anything else.

1. Get to the CLI.
   How to do this depends on the state of the machine. Do not hit the *break key* or *white button* or run the FRUs. Each of these destroy the state of the microcode.

2. Get into the EXPerimental MONitor.

```
CLI> x expmon

EM>
```

3. Stop the machine.

```
EM> SM
```

4. Get the microtrace and copy down the last 10 addresses.

```
EM> trace
        ...
        09   2BOE
        08   2A90
        07   2D8C
        06   2D8D
        05   2D8E
        04   2D8F
        03   3D90
        02   2D91
        01   2D92
        00   2DA7
TRACE>
```

5. Exit the trace program and get to the CLI.

```
TRACE> x
EM> bye
CLI>
```

6. Boot the machine.

```
CLI>boot
```

When/if system gets to EEDB, get the configuration which is running:

7. Find the default configuration.

```
EEDB: show_default

Default configuration is D_10_20_0
```

8. Write down which microcode configuration the machine is running.
It is at the end of the list generated by:

```
EEDB: vd d_10_20_0

Configurations :
D_10_20_0
INITIALIZE.10.0.1D              7/08/88 12:11:51 Key: 14F85804
...
MICROCODE.6.382
EEDB: quit
```

Here the microcode configuration is MICROCODE.6.382

9. Run FRU level 2 on all boards ... note any problems.

10. Run microdiags.

11. Give info collected to DJL for analysis...

♦

INFO: How to run the Configure_Pad program	
Applicable to:	*Fix:*
References:	*Keywords:* Cli, Irae, Pad, X25
Originated from: Ken	*Revised by:*

The CONFIGURE_PAD program is used to configure a Memotec SP-830 PAD for operation with the Series 200 IOC. When the program is started, it will prompt the user with:

```
Options are:
0 => Exit
1 => Manual Configure [ Any port ]
2 => Auto   Configure [ IOC port ]
3 => Auto   Configure [ Environment port, CTS (hardware) flow control ]
4 => Auto   Configure [ Environment port, XON_XOFF flow control ]
Enter option :
```

At this point, no interaction with the PAD has occured and the user may use either of the modes to continue. When manual mode is selected, the user is placed in direct communication through the IOC and is expected to have some knowledge of the PAD SCL (Supervisory Command Language) and of PAD operation in general. To exit manual operation and return to the main menu prompt type [Ctrl]-[C]. When using manual mode, remember that some configuration parameters only take affect after *Warm Start* of the PAD. Also remember that to get the PAD's attention after a *Cold Start* takes the sequence (. . . [CR]) (period, period, period, return). If the service banner does not appear immediately after invoking the manual mode and still does not appear after the ...[CR] sequence, check the cable and PAD. The LED corresponding to the port to which the IOC is connected should be on. If all is in order, the PAD can be powered off then on to force a *Warm Start* and the operation tried again. If there is still no response from the PAD, the user may try a *Cold Start* (the blue button in the back of the PAD), however this will require that all ports be reconfigured.

If `Auto Configure` is selected, the program will prompt the user for several parameters. The program will ask the user if **Verbose** mode is desired. In **Verbose** mode, the dialogue between the IOC and PAD is displayed on the console. After responding to this question, the program will ask for the number of the port which is to be configured. In most cases this is the same as the port to which the IOC is connected. It is possible to configure the other ports if desired from one IOC. The program next asks for the DTE address. This is the address supplied by the X25 carrier, consult with the customer to get this number. Note that this number also needs to be reported to RATIONAL for remote debugging and remote diagnostic support. Finally the program asks for the level 3 packet size (in bytes). After entering this number, the program will set up several parameters in the PAD and end with a *Warm Start* to activate the newly set parameters (*Warm Start* takes about 5 seconds, do not be alarmed at the pause near the end of the setup sequence). If all is successful, the program will announce that the port has been configured. At this point, other ports may be configured if desired. If auto configuration fails, the user may correct the problem by entering manual mode or with a *Cold Start*.

When using CTS flow control, the COMM distribution must have the *Hardware Flow Control* switch(es) properly set. For *Xon_Xoff Flow Control*, be sure the environment has set the terminal port for *Xon_Xoff Flow Control* and that the flow control characters for that port are **DC1** and **DC3**. If other flow control characters are substituted, the PAD must also be configured with the other characters, see the SCL section of the PAD manual and use the manual configuration mode to change the XON and XOFF characters.

Note that these are global characters for the PAD - they take effect for all ports.
♦

INFO: Checkdisk error messages	
Applicable to:	*Fix:*
References:	*Keywords:* Cli, Disk, Ecc Error
Originated from: Gpa, Ken	*Revised by:*

While running:

```
CLI> x checkdisk
```

you get the message:

```
Previously undetected bad block at <location>
Special condition transfer error data check (ECC)
```

Question: Is there any corrective action we should take based on these messages?

Answer: The drive may be going sour. One should check the error logs to see if there are any patterns to the retargetted blocks - same head? same cylinder? same sector accross several cylinders? Make sure cables are in tight at the drives. Schedule a Preventive Maintenance and run `diskx` for 1 hour to be sure no new blocks show up.

Question: What does the second message mean? Is this just a soft ECC error?

Answer: This is the output of the `Image` function on the disk status from the drive error. Four interesting bits were on in the disk status: `Special_Condition`, `Transfer_Error`, `Data_Check`, and `Soft_Ecc_Error`.

♦

TIP: System locks up around CLI level	
Applicable to:	*Fix:*
References:	*Keywords:* Ascii, Boot, Cli, Hang-Up, Terminal
Originated from: Mam	*Revised by:*

When booting from the disk, the system locks up after answering [CR] to the message:

```
Enter name of configuration to boot [STANDARD] :
```

It may be that the console is set for *even* parity rather than *space*.

The character `Ascii.Cr` has the code `10#13#`, i.e. `2#00001101#`. This is odd parity, so to make it even the 8th bit becomes 1, making the character code `2#10001101#` which is unknown to the R1000, since it lies outside the ASCII range.

◆

INFO: How to dump the IOP states	
Applicable to:	*Fix:*
References:	*Keywords:* Crash, Iop, Tape
Originated from: Ken	*Revised by:* Deg, Ken

A crash dump tape contains information about the state of the machine. Part of the crash dump tape contains information about the last two IOP states. Actually, the last five IOP states are stored in `IOP_Dump1..5` files.

If a crash dump cannot be taken or if KEN needs more information than just the last two `IOP_Dumps`, then dumping the `IOP_Dump*` files may be useful. To dump these files on a tape:

```
CLI> Dump/V IOP_DUMP*
```

◆

INFO: System.Assertion_Error at boot time	
Applicable to:	*Fix:*
References:	*Keywords:* Ecc Error, Exception, Kernel
Originated from: Gpa	*Revised by:*

A few seconds after virtual memory starts, you get:

```
Kernel Assert Failure detected by kernel debugger stub. System shutting
down.
Exception : !LRM.SYSTEM.ASSERTION_ERROR, from PC=#199808, #707

Cluster manager error at PC= 3AB8, error code= 0C
Cluster manager detected R1000 Machine Check

...

Do you want a crash dump tape ?
```

If running FRUs and microdiags turn up nothing, then copy down the PC and offset, and ask GPA what to do next. The problem is likely to be corrupted data in BTREE manager. You need to restore from backup tapes.

This problem is almost always caused by overheating problems, multibit ECC errors, or other causes of disk corruption. If possible, investigate these possibilities (examine previous console output, error logs, etc.) before restoring from backup. This problem has not been seen to happen without such a cause in a couple of years.

Since system will not boot past virtual memory one cannot do

```
Kernel: Show_Error_Log
```

since this only works after virtual memory has been started. One test that can be run is:

```
CLI> Checkdisk
```

This may find previously unreported hard disk errors.
♦

INFO: Determining IOP Kernel's version	
Applicable to:	*Fix:*
References:	*Keywords:* Iop, Kernel, Version
Originated from: Deg	*Revised by:*

There are two ways of determining the version of the Kernel:

1. When the machine is running:
   - Put into interactive mode
   - Hit break key
   - Enter debugger; do not leave things in this state since the machine is stopped
   - Type `@400,100'`

     This will display a bunch of characters of which will contain `KERNEL 0.4.17` or something
   - Type [Esc]-[G]. This starts the machine running again. It is a *very important* step.
2. When the machine is down, reboot, and look for message like:

   ```
   INITIALIZING I/O PROCESSOR KERNEL 0.4.17
   ```

In both cases `0.4.17` is the version of the IOP Kernel.
♦

TIP: Kernel assert failure		
Applicable to:	*Fix:*	
References: CSR 3201	*Keywords:* Constraint Error, Crash, Kernel	
Originated from: Cbh, Clp, Vnv	*Revised by:*	

A few minutes after boot had completed, the machine crashed with the following message:

```
Kernel assert failure detected by kernel debugger stub.  System
shutting down.  Exception: Constraint_Error (Variant), from
PC = #19A008, #882
Sequencer has detected a machine check
```

The trace for this crash ended with:

```
0213
0209
0209
```

The FRUS and microdiags were unable to finger the culprit.

It is believed that this type of crash is caused by hardware; the two suspects are the Disk/Tape Controller and disk unit 0. You should check the error logs for possible report of related problems.
♦

HINT: How to destroy a persistent object at EEDB level	
Applicable to:	*Fix:*
References: CSR 3571	*Keywords:* Destruction, Directory, Eedb
Originated from: Mam, Trw	*Revised by:* Mv, Ray

This is a last ditch method for removing an object. It removes the object by by-passing all the usual checks.

1. If the object you want to delete is a view, remove *all* its imports first by using `Cmvc.Remove_Import`.

2. Run the `Directory` daemon and the `Ada` daemon. *If they do not succeed, don't do anything until you have talked to TRW.*

3. Elaborate `Dir_Tester`. At the `EEDB:` prompt at the console, type:

```
EEDB: elaborate dt[CR]
    CG_DIR_TESTS.9.0.0D              5/29/87 16:52:21
    CG_DIR:
```

4. Run `Dir_Tester`. At the `CG_DIR:` prompt, type:

```
CG_DIR: as Cdir[CR]
CDIR_TESTER started
====>> DIR_TEST <<====
---- dirtest directory tester.
Type argument (@argument for indirect):
```

5. Disable access control. At the `Type argument (@argument for indirect):` prompt, type:

```
Type argument (@argument for indirect): /disable_access_control[CR]
Type argument (@argument for indirect):
```

6. Set `Dir_Tester`'s context. At the `Type argument (@argument for indirect):` prompt, type:

```
Type argument (@argument for indirect): /[CR]
COMP_UNIT>
```

   Go to the directory where you want to delete a unit: *Type only* the part of the command in **bold**. The rest is displayed as part of electric completion.

```
COMP_UNIT> Goto Decl named: !MACHINE[CR]
====>> DIR_TEST <<====
MACHINE : WORLD;
VAR>
```

7. Remove the damaged object. First type:

```
VAR> Xecute Directory[CR]
command =
```

Then remove the object (`Temporary` in this example) by typing:

```
command = DM_REMOVE_Child TEMPORARY[CR]
child name: TEMPORARY
status: SUCCESSFUL
VAR>
```

8. Exit

```
VAR> Quit[CR]
Type argument (@argument for indirect):
```

9. Re-enable access control.

```
Type argument (@argument for indirect): /enable_access_control[CR]
Type argument (@argument for indirect):
```

10. Exit the `Dir_Tester`.

```
Type argument (@argument for indirect): /q[CR]
==== dirtest PASSED ================================
     Total passed = 1
====>> Elaborator Database <<=====
CG_DIR: CDIR_TESTER finished
```

Wait until you get the `CG_DIR:` prompt. You may have to hit [Ctrl][Z] and [CR] a few times. At the `CG_DIR:` prompt, type:

```
CG_DIR: quit[CR]
EEDB:
```

At the `EEDB:` prompt, type:

```
EEDB: unelaborate dt[CR]
Subsystem: [CR]
Unelaborated CG_DIR_TESTS.9.0.0D
EEDB:
```

This completes the surgery. Back in the environment, refresh the directory listing and verify that the object has disappeared from the display. Run the following utilities and check that they all completed successfully:

1. In the directory just repaired, run `Repair_Directory`.

2. Re-run the `Directory` and Ada daemons.

Note: `Repair_Directory` might report some anomalies, if so, running it a second time ensures that everything is fixed.

♦

TIP: Reclaiming old configuration/subsystem space	
Applicable to:	*Fix:*
References:	*Keywords:* Eedb, Garbage Collection
Originated from: Rjg	*Revised by:*

You can reclaim space by deleting old environment subsytems. The amount you reclaim is entirely dependent on the amount of old subsystems you have lying around. For example, on *Gator* it was possible to reclaim around 1500 blocks (= 1.5MB), a significant (isn't 1% significant?) savings.
Here's how (note that all these commands are done from EEDB):

1. Delete any old unwanted configurations
   Use the `Display` command to list configurations ("+" is a multiple character matching wildcard):

   ```
   EEDB: Display +
   ```

   Use the `Delete` command on old configuration(s):

   ```
   EEDB: Delete D_9_25_1
   ```

2. Check to make sure that no configuration names contain a "."
   This is to ensure the following step doesn't delete any configurations. Use the `Display` command again.

3. Delete unused subsystems
   Use the `Delete` command, matching all subsystems (they always contain a "."):

   ```
   EEDB: Delete +.+
   ```

   This is OK because it will not delete subsystems that are part of a configuration. There will be a bunch of warning messages regarding subsystems that can't be deleted.

4. Delete code segments no longer associated with subsystem versions
   Use the `Reclaim_Space` command:

   ```
   EEDB: Reclaim_Space
   ```

   It will tell you how many blocks were reclaimed.

♦

TIP: Promoting an Ada unit using Cg_Dir_Tests.	
Applicable to:	*Fix:*
References:	*Keywords:* Eedb, Login, Promote
Originated from: Mam	*Revised by:*

If you can't login into a system and you need to promote an object such as `!Machine.Initialize` or the `>>Terminal_Type<<_Commands` procedures you can use `dt`.

1. Elaborate `dt` at the `EEDB`. If it doesn't elaborate you will probably have to rebuild it.

```
EEDB: e dt
CG_DIR_TESTS.9.0.0D            5/29/87 16:52:21
CG_DIR:
```

2. Start the `CDIR_TEST` program

```
CG_DIR: run cdir
CDIR_TESTER started

=====>> DIR_TEST <<=====
Type argument (@argument for indirect):
```

3. Turn off access control.

```
---- dirtest directory tester.
Type argument (@argument for indirect): /disable_access_control
Type argument (@argument for indirect):
```

4. Enter the Universe.

```
Type argument (@argument for indirect): /
COMP_UNIT>
```

5. Go to the unit that you wish to promote.
Type only the part of the command in **bold**. The rest is displayed as part of electric completion.

```
COMP_UNIT> Goto Decl named: !MACHINE.INITIALIZE'BODY[CR]
procedure INITIALIZE is separate;
```

6. Promote the unit.

```
SUBPROGRAM_BODY> Make Promotion
goal state = Coded
100  instructions for subprog INITIALIZE
434  instructions for segment 147712
results: SUCCESSFUL
```

7. Exit the Universe.

```
SUBPROGRAM_BODY> Quit
```

8. **Turn on Access Control**

```
Type argument (@argument for indirect): /enable_access_control
```

9. **Exit**

```
Type argument (@argument for indirect): /q
==== dirtest PASSED ====================
 Total passed =  1

=====>> Elaborator Database <<=====
CG_DIR: CDIR_TESTER finished

CG_DIR: quit
```

10. **Unelaborate** dt

```
EEDB: un dt
Subsystem: [CR]
Unelaborated CG_DIR_TESTS.9.0.0D
EEDB:
```

♦

INFO: R1000_Code_Generator.Initialize fails at boot time		
Applicable to:	*Fix:*	
References:	*Keywords:* Archive, Boot, Code Generator, Debugger, Eedb, Exception, Internal Error	
Originated from: Jim, Rcp, Rjg, Swb	*Revised by:*	

During boot, the following messages are displayed on the console:

```
...
EEDB elaborated Native_Debugger.10.4.0D
Debugger_Elaboration Problem_registering_Types
exception <Unit = 4804117, Ord = 1 >, from PC=#346C14,#A1
...
Boot running R1000_Code_generator.Initialize
Boot unexpected_exception R1000_Code_Generator.Initialize failed:
  exception: <Unit=4804117, Ord = 1>, from PC=#346C14,#A1

Boot running Check_Device_Declarations
...
Boot running "!Machine".Initialize
Program Internal_Error assembly error: code segment overflow, program too
large.
```

At this point only port 16 and the console are alive. It is possible to log in and run some commands, but only interpreted commands. Any commands that must go through the code generator fail with the following message:

```
Program Internal_Error assembly error: code segment overflow, program too
large.
```

This is probably caused by missing files in `!Machine.Cg_Data`; most likely the customer inadvertently deleted them. There is no effect until the next reboot. Normally the directory looks like this:

```
!Machine.Cg_Data
   Forms_Tab15
   Instructions_Spec
   Range_Tab15
   Value_Tab15
```

You cannot use `Archive` commands on the affected R1000 to restore the files, because `Archive` commands are fed through the code generator. The machine would probably recover as long as `Instructions_Spec` were restored, but it is much too long to easily type in.

- If the machine is on a network with another R1000, use `Archive` commands on an unaffected R1000 to "push" the files onto the affected machine. This will only work if the Archive Server is alive. Then, take a snapshot and reboot the machine.

- Use `Dir_Tester` from the EEDB to read an `Archive` tape. Here's how:

  1. Make `Archive` tape of `!Machine.Cg_Data` on another machine.

2. Elaborate dt at the EEDB level.

```
EEDB: e dt
CG_DIR_TESTS.9.0.0D
CG_DIR:
```

3. Turn off access control

```
Type argument (@argument for indirect:) /disable_access_control
```

4. Enter the Universe

```
Type argument: (@argument for indirect): /
```

5. **Note:** This is an electric completion interface, type slowly and stop typing when it fills out the rest of the command. The part you have to type is in **bold**:

```
COMP_UNIT> Xecute Directory_command [CR]
command = SA_Restore [CR]
objects (<return> = ??) = [CR]
use_prefix (<return> = *) = [CR]
for_prefix (<return> = *) = [CR]
options (<return> = r1000) = [CR]
device (<return> = tape) = [CR]
log = <Some file name> [CR]
```

6. Load Archive tape, answer mount request.

7. Exit the Universe

```
COMP_UNIT> Quit
```

8. Turn on access control

```
Type argument (@argument for indirect): /enable_access_control
```

9. Exit

```
Type argument (@argument for indirect): /q
CG_DIR: CDIR_TESTER finished
CG_Dir: quit
```

10. Unelaborate dt

```
EEDB> un dt
```

11. Take a snapshot

```
EEDB> snap
```

12. *White button* the machine.

♦

TIP: Finding out what subsystem raised an exception	
Applicable to:	*Fix:*
References:	*Keywords:* Eedb, Exception, Subsystem
Originated from: Unknown	*Revised by:*

When you get an unknown exception such as:

```
Execution.Execute_Command message - Unexpected exception: <Exception: Unit =
4649814, Ord = 14>, from PC = #FE100B, #198
```

you may determine what subsystem raised this exception. At EEDB, find what subsystem contains this PC by typing:

```
EEDB: find 16#FE100B#
```

It returns you a message like:

```
DIRECTORY.10.2.1D
```

◆

INFO: Building a configuration	
Applicable to:	*Fix:*
References:	*Keywords:* Configuration, Eedb, Subsystem
Originated from: Phl	*Revised by:*

If you want to use EEDB-level tools such as `dt` or `ed`, it may happen that when you do:

```
EEDB: elaborate dt
```

you receive the message:

```
A subsystem with name NETWORK is already elaborated
```

This is because the configuration you elaborate (in this example `dt`) was built on a configuration different from the running one.

The solution is to rebuild a configuration based on the running one.

1. Do:

   ```
   EEDB: vdisplay dt
   ```

   and note the two first lines of the display:

   ```
   CG_DIR_TESTS.9.0.0D
   R1000_CODE_GEN.10.3.9D
   ```

2. Delete the existing `dt`:

   ```
   EEDB: delete dt
   ```

3. Determine the running configuration:

   ```
   EEDB: running
   ```

   This tells you something like:

   ```
   D_10_20_0_MAIL
   ...
   ```

4. Re-create a configuration called `dt`, based on the running configuration; the parent subsystem is the one specified be the second line of `vd`'s display (see 1 above).

   ```
   EEDB: build_configuration
   New Configuration: dt[CR]
   Existing Configuration: D_10_20_0_MAIL[CR]
   Parent subsystem: R1000_CODE_GEN[CR]
   Subsystem.Version: [CR]
   ```

5. Add the subsystem containing the tool you want to activate. This subsystem is the one specified by the first line of `vd`'s display (see 1 above).

```
EEDB: add
Existing Configuration: dt[CR]
Subsystem.Version: CG_DIR_TESTS.9.0.0D[CR]
```

6. You may now elaborate the new `dt` by typing:

```
EEDB: elaborate dt
```

♦

INFO: Creating a user from EEDB	
Applicable to:	*Fix:*
References:	*Keywords:* Eedb, Password, User
Originated from: Phl	*Revised by:*

Creating a user from EEDB may for instance be useful if the operator password has been forgotten, or if some password policy problem prevents users from logging in.

1. Elaborate ed:

   ```
   EEDB: elaborate ed
   ```

2. Create the user:

   ```
   ED: x create_user
   ```

   and answer the password and username questions.

3. Exit ed:

   ```
   ED: quit
   ```

4. Unelaborate ed:

   ```
   EEDB: unelaborate ed
   ```

♦

TIP: Location and size of managers' state files	
Applicable to: D1, D2	*Fix:*
References:	*Keywords:* Disk Space, Eedb, Object Manager, State
Originated from: Cbh, Jim, Mv	*Revised by:*

This card describes how to figure out on which volume the object managers' states are kept what their size is (state files belong to the polymorphic file system managed by EEDB; they are *not* Environment files). There are two reasons why you may want to obtain this information:

- If all the state files are allocated to the same volume in a multi-volume system with evenly distributed data, this volume is more likely to experience garbage collection problems than the rest. This is because object managers generate garbage on their volumes with each snapshot (we have in the past shipped systems with all object managers on volume 1...).
- When adding an additional volume to a system, the restored backup will not be distributed evenly over the N+1 volumes In fact, `Backup` tries hard not to redistribute the space: the added volume will be almost empty. If the state files can be moved to this volume, some 40 Mb (typical) of free space is added to the other volumes.

In both case, you can redistribute state files by using procedure `Operator.Archive_On_Shutdown`.

The procedure to find out the size and location of the state files is as follows:

1. Elaborate the `OM` configuration:

```
EEDB: elaborate om
OM_TESTS.9.0.0D                 2/06/87 22:41:08
```

2. Display all state files:

```
Om_Tests: VDIR *.STATE

[16#104#, DATA, 16#0DC32C#]       27192  NULL_DEVICE.state
[16#104#, DATA, 16#0DC32A#]      122743  GROUP.state
[16#103#, DATA, 16#279D7C#]      187542  PIPE.state
[16#104#, DATA, 16#0DC330#]      486724  ARCHIVED_CODE.state
[16#105#, DATA, 16#007D6C#]    93086095  ADA.state
[16#105#, DATA, 16#007D67#]      196386  TERMINAL.state
[16#105#, DATA, 16#007D69#]       29332  TAPE.state
[16#102#, DATA, 16#308034#]     2555378  LINK.state
[16#102#, DATA, 16#30802C#]      226094  SESSION.state
[16#102#, DATA, 16#308030#]       43029  PROGRAM_LIBRARY.state
[16#103#, DATA, 16#279D7E#]    53101174  FILE.state
[16#102#, DATA, 16#308032#]     7841409  DDB.state
[16#104#, DATA, 16#0DC32E#]       60852  CONFIGURATION.state
[16#102#, DATA, 16#30802E#]    69247879  DIRECTORY.state
[16#103#, DATA, 16#279D7A#]      259097  USER.state
[16#105#, DATA, 16#007D65#]    25461415  CODE_SEGMENT.state
```

The listing contains the size of each manager (state file) in bits. The first hexadecimal number is the VP of the manager. This number will be used to determine which volume the manager state is allocated to.

3. Exit from Om_Tests:

```
Om_Tests: QUIT
```

4. Unelaborate the OM configuration:

```
EEDB: UNELABORATE OM
Subsystem:
Unelaborated OM_TESTS.9.0.0D
```

5. Enter privileged mode at the Kernel level:

```
Kernel: ENABLE_PRIV_CMDS
You are enabling a set of commands which must be used
with extreme care.  They should be used only by
knowledgeable support personnel.  These commands can
easily crash/hang the machine; some can competely trash
the state of the machine such that you must recover the
machine from backup tapes.
Proceed [FALSE]: YES
Password: SECRET
```

6. List all VPs:

```
*Kernel: SHOW_VPS

(VP => 4, VOLUME => 1)
(VP => 5, VOLUME => 1)
(VP => 6, VOLUME => 1)
(VP => 7, VOLUME => 1)
    .
    .
    .
(VP => 257, VOLUME => 2)
(VP => 258, VOLUME => 3)
(VP => 259, VOLUME => 1)
(VP => 260, VOLUME => 2)
(VP => 261, VOLUME => 4)
```

This is a truncated listing. Only the last VP numbers are of interest.

7. Disable privileged mode:

```
*Kernel: DISABLE_PRIV_CMDS
```

At this point, you have enough information to determine the following:

Manager	Size (bits)	VP (hex/decimal)	Volume
ADA	93086095	16#105# / 261	4
ARCHIVED_CODE	486724	16#104# / 260	2
CODE_SEGMENT	25461415	16#105# / 261	4

```
CONFIGURATION          60852   16#104# / 260       2
DDB                  7841409   16#102# / 258       3
DIRECTORY           69247879   16#102# / 258       3
FILE                53101174   16#103# / 259       1
GROUP                 122743   16#104# / 260       2
LINK                 2555378   16#102# / 258       3
NULL_DEVICE            27192   16#104# / 260       2
PIPE                  187542   16#103# / 259       1
PROGRAM_LIBRARY        43029   16#102# / 258       3
SESSION               226094   16#102# / 258       3
TAPE                   29332   16#105# / 261       4
TERMINAL              196386   16#105# / 261       4
USER                  259097   16#103# / 259       1
```

Note: This procedure is accurate for D1, but not for D2. On a D2 system, each object manager gets its own VP somewhere in the range 1001 .. 1022, and the mapping is fixed. This change actually makes the problem easier. The D2 mapping is:

```
ADA                 1001
DDB                 1002
FILE                1003
USER                1004
GROUP               1005
SESSION             1006
TAPE                1007
TERMINAL            1008
DIRECTORY           1009
CONFIGURATION       1010
CODE_SEGMENT        1011
LINK                1012
NULL_DEVICE         1013
PIPE                1014
ARCHIVED_CODE       1015
PROGRAM_LIBRARY     1016
```

◆

INFO: Functional Diagnostics does not halt on UPC 0103	
Applicable to:	*Fix:*
References:	*Keywords:* Board, Cli, Diagnostic, Microcode, Preventive Maintenance, Seq
Originated from: Jaw, Mam, Pam	*Revised by:*

When running Functional Diagnostics, you expect to see:

```
HALT - NEXT UPC IS 0103
NO ECC ERRORS HAVE BEEN DETECTED SINCE THE LAST BOOT
```

which contains the magic number 0103.

If you don't get this magic number, here are some possibilities:

- if the machine is not booted to DIAG (which loads diagnostics microcode) the message:

  ```
  HALT - NEXT UPC IS 0101
  NO ECC ERRORS HAVE BEEN DETECTED SINCE THE LAST BOOT
  ```

  will be diplayed.

- If you get:

  ```
  HALT - NEXT UPC IS 22A3
  NO ECC ERRORS HAVE BEEN DETECTED SINCE THE LAST BOOT
  ```

  there are chances (90%) that the SEQ board is bad. Other possibilities include MEM and IOC.

◆

INFO: Diskmd, a DFS-based disk utility	
Applicable to:	*Fix:*
References:	*Keywords:* Cli, Dfs, Diagnostic, Disk
Originated from: Mam	*Revised by:*

To enter `Diskmd`:

```
CLI> x diskmd
DISKMD>
```

The `Diskmd` program is a DFS based disk utility which is useful for a wide range of disk problems. It is user interface-based and has commands that allow low-level operations with disk drives. The program has two 1024 byte (R1000 disk block size) buffers, a read and a write buffer.

Commands:

`BYE` Exits DISKMD.

`COPY` Copies the contents of the read buffer into the write buffer.

`CSTATUS` Displays present status of current unit.

`CTS` The CTS command requires a single decimal argument which is converted into cylinder, track, and sector and then inserted into the macro evaluation buffer. For example, to read the first page of bad block information do *rd [cts 5]*.

`DATA` The DATA command requires a single hex argument which is truncated to 16 bits and used to fill the write buffer.

`DBN` The DBN command requires three decimal arguments which are converted into a disk block number and inserted into then macro evaluation buffer.

`DISPLAY` The DISPLAY command displays the contents of the read buffer.

`EDIT` The EDIT command requires two hex arguments. The first is used as an address in the write buffer and the second is a 16 bit data word which is inserted into the write buffer at the address specified.

`FORMATS` Format Sector requires three decimal arguments : cylinder, head and sector. It then formats the single specified sector with good user and manufacturer flags and data from the first half of the write buffer.

`HELP` Displays this text on the console.

`RD` The RD command requires three decimal arguments which are the cylinder, track, and sector of a 1KB disk block to be read from the current unit into the read buffer.

If any errors are encountered during the read the buffer will only contain data from sector successfully read, the other half of the read buffer will remain unchanged.

RS — Like RD above only it reads a single 512 byte sector into the first half of the read buffer.

SEEK — The SEEK command requires a single decimal argument, the cylinder number. The heads of the currently selected drive are positioned to that cylinder but no transfer is done.

STATUS — Displays the status from the last error for the current unit. If no errors have occurred the displayed status is invalid.

UNIT — The UNIT command requires a single decimal argument which is used for all disk specific I/O from then on. The UNIT command software write protects the newly selected disk until the WRENABLE command is issued.

VERIFY — Compares the read buffer with the write buffer. Differences are displayed on the console.

WR — The WR command requires three decimal arguments, a cylinder, track, and sector number. A 1KB transfer from the write buffer to that sector is stored on the current unit. If any errors occur the transfer is aborted. Doesn't write header.

WRENABLE — The WRENABLE command is issued to write-enable the current unit. It remains in effect until the UNIT command is issued.

♦

INFO: Analyzing a hardware crash	
Applicable to:	*Fix:*
References:	*Keywords:* Board, Cli, Crash, Diagnostic
Originated from: Unknown	*Revised by:*

When the R1000 CPU stops running, the IOP queries all the boards to determine what caused it to stop.

The IOP then runs a program called EMBALM which attempts to determine in more detail what went wrong. This program is very good at analyzing WCS parity errors, but usually is not helpful if the failure was elsewhere. Therefore, it is important to manually stop the machine and note the cause of the error.

The following is the procedure to manually stop the machine. All possible errors are listed here, with suggested responses given where appropriate.
It is important to note that the board reporting the error is not necessarily the board that failed.
The machine crashes:

```
DO YOU WANT TO RUN THE HARDWARE DIAGNOSTIC [Y] :  N

CLI> X EXPMON

EM> THERE ARE 4 8 MEGABYTE MEMORY BOARDS FOR A TOTAL OF 32 MEGABYTES
EM> SM
```

Some error messages will then be output, here are the possibilities where similar errors can happen on multiple boards, they are listed together with a note on which boards could report the error.
Note: MEM*i* means MEM0 or MEM1 or MEM2 or MEM3

- BACKPLANE BUSSES.

  ```
  ADDRESS BUS PARITY ERROR:  FIU, MEMi
  ```

  The address bus had bad data. If all possible boards report this, then the problem is with the source of the address bus. Possible address bus sources are: VAL, TYP, SEQ, FIU. Determining the source involves listing and interpreting the microcode trace, and is beyond the scope of this card. If only one board reports the error, it is the culprit. If some but not all report it, then something weird is happening and you should not suspect hardware (could be initialization problem, satanic influence, etc.).

  ```
  MICRO ADDRESS BUS PARITY ERROR: SYS, IOC, VAL, TYP, FIU
  ```

  The micro address bus had bad data. If all possible boards report this, then the problem is with the SEQ board. If only one board reports it, then it's bad, else something weird is happening.

- FOREPLANE BUSSES.

  ```
  VAL BUS PARITY ERROR: SEQ, FIU, IOC, SYS
  ```

  The VAL bus had bad data. If all possible boards report this, then the problem is with the source of the bus. Possible sources are:
  VAL, SEQ, FIU, SYS, IOC, MEMi.

Determining the source involves listing and interpreting the microcode trace, and is beyond the scope of this card.
If only one board reports the error, it is the culprit. If some but not all report it, then something wierd is happening.

```
TYP BUS PARITY ERROR: SEQ, FIU, IOC, SYS
```

The TYP bus had bad data. If all possible boards report this, then the problem is with the source of the bus. Possible sources are:
TYP, SEQ, FIU, SYS, IOC, MEMi.
Determining the source involves listing and interpreting the microcode trace, and is beyond the scope of this card.
If only one board reports the error, it is the culprit. If some but not all report it, then something weird is happening.

```
FIU BUS PARITY ERROR: SEQ, FIU
```

The FIU bus had bad data. If all possible boards report this, then the problem is with the source of the bus. Possible sources are:
VAL, TYP, SEQ, FIU
Determining the source involves listing and interpreting the microcode trace, and is beyond the scope of this card. If only one board reports the error, it is the culprit. If some but not all report it, then something weird is happening.

- WCS ERRORS.

```
WCS PARITY ERROR: VAL, TYP, SEQ, FIU, SYS, IOC
```

The Writeable Control Store RAM has failed on the reporting board, and it is the culprit.
Note: on Series 100, sometimes this is caused by improper loading of the RAMs, and is actually a problem with the SYS or IOA. Run P2SYS before swapping a board.
Some old boards (serial number in low 20's or lower) have INMOS 1400 RAMs. These are notoriously unreliable, and these boards should be kept from production machines whenever possible.

- HALT: SEQ.

If this happened, note the address. This is either a microcode error or a software error.

- NO MACHINE CHECKS DETECTED.

The machine was running, but maybe hung. This is either a microcode error or a software error.

- INTERNAL ERRORS.

```
MEMi TAGSTOREi PARITY ERROR: MEMi
```

The memory board reporting the error is probably the culprit. The only other possibility is that the tagstore is being written, and the VAL bus data is bad. In this case other boards should report a VAL BUS PARITY ERROR, so ignore the memory error.

```
A_BUS PARITY ERROR:  VAL, TYP
```

This is probably a AMD 2149 failure and the reporting board is bad.

```
B_BUS PARITY ERROR:  VAL, TYP
```

This is probably a AMD 2149 failure and the reporting board is bad.

```
C_BUS PARITY ERROR: VAL, TYP
```

This could be a FIU bus problem, in which case some other board may report a problem. If not, suspect the reporting board.

```
DECODE RAM PARITY ERROR: SEQ
```

This is probably a AMD 2149 failure and the reporting board is bad.

```
SCAVENGER RAM PARITY ERROR: FIU
```

This is probably a AMD 2149 failure and the reporting board is bad.

```
REFRESH MACHINE CHECK: FIU
```

This is usually a microcode error.

- SYSBUS ERRORS.

```
TRANSFER HARD ERROR: SYS

BID ERROR: SYS

MICRO BREAK EVENT: SYS
```

◆

INFO: MT, a DFS-based tape utility	
Applicable to:	*Fix:*
References:	*Keywords:* Cli, Dfs, Diagnostic, Tape
Originated from: Ken	*Revised by:*

To enter MT:

```
CLI> x mt
MT>
```

The MT program is a DFS based magnetic tape utility. It is used to transfer DFS files between a DFS disk and an "MT" format tape. The MT program utilizes the macro user interface.

Command:

LOAD The LOAD command is used to transfer files from a tape to a DFS disk. The LOAD command requires no arguments and will transfer all files on the tape. The LOAD command leaves the tape positioned at EOT.

Switches:

/F= The /F switch allows the user to enter a single file name specifier that will be used to selectively load files from the tape. Normal wildcard characters are allowed. The default is /F=*, i.e. all files. If used, the /F switch must be the last one in the command line.

/I Like the /V switch below however only announces filenames that match the file name specifier (see /F switch).

/N The /N switch causes the LOAD command to read the files from tape but not write them to disk. This switch may be used in conjunction with the /V switch to see what's on a tape or to position the tape at EOT to enable appending files to an existing tape.

/Q The /Q switch will query the user for file overwrite permission. Default is not to query.

/V The /V switch causes the MT program to display the name of each file read from tape. This will override the /I switch is both are specified.

/UNLOAD The /UNLOAD switch causes the LOAD command to unload the tape rather than leaving the tape positioned at EOT.

/UNIT= The /UNIT= switch may be used to specify a specific

```
        tape unit for the transfer.  If the /UNIT= switch is
        not present, unit zero will be used.
```

Examples:

```
MT> load/v/unload/unit=1
```

Loads all files from tape, announces files loaded then unloads the tape. Uses drive 1.

```
MT> load/i/f=*.cli
```

Loads all files that end in `.cli`. Announces those files loaded.

```
MT> load/i/q/f=*.cli
```

Like above with the added feature of asking user for overwrite permission if the already exists on the disk.

```
MT> load/i/n/f=*junk*
```

Lists all files that have the string "junk" their filename.

Command:

```
DUMP    The DUMP command is used to transfer files from a DFS
        disk to a tape.  The DUMP command will leave the tape
        positioned at EOT upon transfer of the last file.  The
        dump command requires at least one argument.  This
        argument is the file specification to dump to tape.
        More than one file specification may be present on the
        command line.  They will be processed in the order
        they appear.
```

Switches:

```
/GCR    The /GCR switch will force the file to be written in GCR
        format if the tape drive supports remote density
        selection.  The /GCR switch may only be used if the
        tape is at BOT.  GCR format records at 6250 bytes
        per inch.

/PE     The /PE switch will force the file to be written in PE
        format if the tape drive supports remote density
        selection.  The /PE switch may only be used if the
        tape is at BOT.  PE format records at 1600 bytes
        per inch.

/LONG_GAPS  The /LONG_GAPS switch will force the tape to be written
        with long or variable interrecord gaps if the tape drive
        supports remote gap selection.  This will increase tape
        throughput.

/SHORT_GAPS The /SHORT_GAPS switch will force the tape to be written
        with short interrecord gaps if the tape drive supports
        remote gap selection.  This will increase tape capacity.
```

```
/THRESHOLD= The /THRESHOLD switch helps the MT program optimize
        throughput.  It requires a single argument.  If a
        file being DUMPed is larger than the THRESHOLD that
        file will be dumped in high speed mode.  If the file
        is smaller it will be dumped in low speed mode.  The
        MT program uses reasonable defaults but the /THRESHOLD
        allows the user to optimize further.

/UNIT=      The /UNIT= switch may be used to specify a specific
        tape unit for the transfer.  If the /UNIT= switch is
        not present, unit zero will be used.

/UNLOAD     The /UNLOAD switch causes the LOAD command to unload
        the tape rather than leaving the tape positioned at
        EOT.

/V      The /V switch causes the MT program to display the
        name of each file read from tape.
```

Examples:

```
MT> dump/v/gcr/threshold=25/short_gaps file1,file2,*file3*
```

Command:

```
REWIND  The REWIND command will reposition the tape to BOT.
```

Switches:

```
/UNIT=  The user may select a unit with the /UNIT= switch.
        By default unit zero is rewound.
```

Examples:

```
MT> rewind/unit=2
```

Command:

```
UNLOAD      The REWIND command will reposition the tape to BOT and
        then unload the tape from the drive.
```

Switches:

```
/UNIT=  The user may select a unit with the /UNIT= switch.
        By default unit zero is unloaded.
```

Examples:

```
MT> unload/unit=3
```

♦

INFO: Unable to log in via the Console Command Interpreter	
Applicable to:	*Fix:*
References:	*Keywords:* Acl, Console Command Interpreter, Group, Login
Originated from: Mam	*Revised by:*

The Console Command Interpreter is launched by `!Machine.Initialize_Servers` and runs with identity `System`. Thus, to be able to log in via the Console Command Interpreter, the group `System` must have read access to `!Users.<Username>`. If read access is not granted to `System`, you will get the message:

```
Invalid User Identity <User>.<Session>.
```

◆

INFO: Unable to log in via the Console Command

BUG: Wrong completion		
Applicable to: D1,D2 and probably D3		*Fix:*
References:	*Keywords:* Completion	
Originated from: Jim	*Revised by:*	

There is a bug such that `Common.Complete` produces an erroneous result when an ambiguous expression is used for an object enclosing the object you want to complete.

For example, I did the following in a command window:

- Typed in `Cmvc@.Expu` and then hit [Complete]. This gave me a completion of `Cmvc.Expunge_Database ...`. This is incorrect and will not semanticize. `Expunge_Database` is located in package `Cmvc_Maintenance`, not `Cmvc`!
- Typed [Object]-[U] twice to get back to the command window containing `Cmvc@.Expu` and hit [Complete] again - this time completion failed!
- Edited the line to read `Cmv@.Expu` and hit [Complete]. This time I got a completion window offering all the package names names that begin with `Cmvc` whether or not they declared an `Expu@`.

This is *a known embarassment, but hard to fix.* There is no outlook as to when it may be fixed.

♦

INFO: Core Editor keymaps implementation	
Applicable to:	*Fix:*
References:	*Keywords:* Core Editor, Keymap, Terminal
Originated from: Deg	*Revised by:* Jls, Lhrp

1. Introduction

This document describes the mechanisms utilized by the Rational Environment Core Editor for handling various terminal types and interpreting special character sequences (escape sequences) as a request for command execution. This mechanism has been developed to allow modification to the existing keymaps as well as introducing user-defined keymaps.

The files used by the Core Editor for Keymap Implementation are:

```
!Machine.Editor_Data.>>Terminal_Type<<_Keys
!Machine.Editor_Data.>>Terminal_Type<<_Key_Names
!Machine.Editor_Data.>>Terminal_Type<<_Commands
!Machine.Editor_Data.Terminal_Types
!Machine.Editor_Data.Terminal_Recognition
!Machine.Editor_Data.Key_Cache
```

Where `>>Terminal_Type<<` is replaced by the particular terminal type of interest. For example Rational, Facit, or VT100 are 3 terminal types defined by Rational.

2. Theory

As a screen oriented editor, the Core Editor must be able to interpret the sequence characters it is receiving, perform the requested operation and then send the appropriate character sequence back to the terminal. There are actually two devices that come into play when we speak about the terminal. First the input device, i.e. the keyboard, and the output device, i.e. the screen. The editor must know how to interpret a given sequence of characters but it must also know what sequence of characters are used to perform the screen manipulations desired. This would include setting scroll regions and drawing the window boundaries.

The Rational Environment provides the mechanism for establishing new terminal types for input only. The device must be able to utilize the ANSI standard escape sequence for screen handling. When a key is typed on a terminal keyboard, an escape sequence of characters is produced. This escape sequence is captured by the Core Editor and mapped to an enumeration value. The enumeration value is then mapped to an Ada command.

3. `>>Terminal_Type<<_Keys` File Theory

This text file contains the mapping of escape sequences to a string value that corresponds to an enumeration value in `Terminal_Key_Names`. This mapping is contained within tables located in `Key_Cache`. During initialization of a machine, tables are built by using the information in the `>>Terminal_Type<<_Keys` file. The number at the beginning of the file tells the environment the number of the table that can be generated (more later). All escape sequences must be preceded by a number describing how many characters are in the escape sequence.

Initially, a table is built (indexed by ASCII characters) containing either a pointer to another table or a string value. Each escape sequence is read from the >>Terminal_Type<<_Keys file. Each character of the escape sequence, starting from the left, is used as the index to the corresponding table. If the next character in the escape sequence is a space, then the table element will contain a string value. If a non space character follows the character, then the table element contains a pointer to another table. That table is used for the next character and so on until the enumeration value is obtained.

4. **Example**

```
4
3[Esc]"@ S_F1
3[Esc]"A S_F2
```

This example shows that at the most, 4 tables can be generated. Each escape sequence has 3 characters. When the S_F1 key is pressed, the environment receives the [Esc]"@ escape sequence. The [Esc] character is looked up in table 1, since the next character is a non space character the element points to another table (table 2). The " character is looked up on table 2 and since the next is a non space character the element points to table 3. The @ character is looked up in table 3 and since the next character *is* a space then the element contains a string value (S_F1). The key name given must be unique and must exist in the >>Terminal_Type<<_Key_Names enumeration type. Also, no escape sequence may be a prefix for another single key escape sequence. In the above example, an escape sequence [Esc]" for F1 would be illegal since it is the prefix for [Esc]"@ and [Esc]"A. This kind of error will prevent installation of the corresponding keymap.

Note: The limit to the number of tables that each Terminal_Keys can specify is 511. If enough tables are not specified then an error will be logged into the system error log and the same keys will not be defined.

The table for ASCII characters are automatically created so if your terminal generates the ASCII character instead of an escape sequence for a key, then the ASCII character is displayed.

5. **>>Terminal_Type<<_Key_Names File Theory**

This Ada specification contains an enumeration type that defines a literal for every key on the keyboard. There should be a literal defined for every escape sequence associated string value specified in >>Terminal_Type<<_Keys if the key is to be defined.

For convenience, constants can be defined to provide aliases for literals. These literals are used by >>Terminal_Type<<_Commands to identify what commands should be executed.

6. **>>Terminal_Type<<_Commands File Theory**

This Ada procedure resolves what Ada commands are bound to enumeration values defined in >>Terminal_Type<<_Key_Names and what mode should be used for execution of the command. There are three modes of execution defined by this procedure.

7. **Modes**

- Interrupt
  Will interrupt a currently executing command so that the specified command can execute.
- Prompt
  Will bring up the specified command into a command window.

- Execute
  Will execute the command when resources are available.

8. **`Terminal_Types` File Theory**

Each terminal type defines six characteristics of the terminal device:

- `Output_Type`
  Specifies the output driver for producing output. The Rational currently supports four output drivers: Rational, Facit, VT100 and XTerm.

- `Input_Type`
  Specifies the name of the terminal type - for example, PCAT (i.e. IBM PC/AT). The terminal type can define its own `>>Terminal_Type<<_Keys`, `>>Terminal_Type<<_Key_Names`, and `>>Terminal_Type<<_Commands` files. If the terminal type does not have these files then the files corresponding to the specified `Output_Type`.

- `Login_Name`
  The terminal type entered at login time to select one of the lines in the `Terminal_Type` file. The default for this field is the `Input_Type`.

- `User_Input`
  This one is used to look for `>>User_Input<<_Commands` in the users login directory. Such a procedure would override parts of the corresponding `!Machine.Editor_Data.@_Commands` procedure. The default for this field is `Input_Type`.

- `Lines`
  Specifies the number of lines on the output device.

- `Columns`
  Specifies the number of columns on the output device.

Rational predefines these characteristics for each of the currently supported output drivers. For user-defined terminal types there must be an entry in the `Terminal_Types` file defining them. The `Terminal_Types` file is read during system initialization.

The syntax for the `Terminal_Types` file is:

```
[Login_Name:] Input_Type [.User_Input] [Output_Type] [Lines [Columns]]
```

If `Output_Type` is omitted then the `Input_Type` must be one of the predefined terminal types (Rational, Facit, VT100, Xterm). If lines or columns is omitted, the default value for the `Input_Type` is used.

A user can explicitly set the terminal type when login in. This can be accomplished by typing an = at the user name prompt and specifying the `>>Terminal_Type<<` at the terminal type prompt. The default terminal type value shown in parentheses is set to the terminal type of the previous session last used on the port.

```
enter user name: =
enter terminal type (VT100): PCAT 72 120
```

The >>Terminal_Type<< specified at the terminal type prompt must have the corresponding keymap files installed or an error message will be displayed.

9. **Terminal_Recognition File Theory**

When a user logs in the Terminal_Type being used needs to be established. This can be done explicitly as described above or automatically by using a Rational defined terminal type or providing a user defined terminal type recognition sequence in the Terminal_Recognition file. When a login occurs the ANSI escape sequence for terminal identification ESC[0c is sent to the terminal. The terminal then responds with a recognition escape sequence that defines the terminal type. First, the recognition sequence is checked to see if it is one of the predefined Rational Terminal types, then the Terminal_Recognition file is searched for recognition sequence. Definitions appearing later will override earlier ones, so a user defined recognition sequence can override predefined Rational terminal types.

Caution: The Facit terminal recognition is the same as a VT100. When a VT100 recognition sequence is received another prompt about the Twist option is sent. If a reply occurs then it is a Facit. A VT100 recognition sequence defined in the Terminal_Recognition file will prevent a Facit from being recognized.

Terminal_Recognition file syntax is:

```
>>Terminal_Type<< Recognition_Sequence
```

For Example:

```
VT100 [Esc][?1; c
RATIONAL [Esc][?9c
```

The space after the semicolon in the VT100 recognition sequence is a wild card for any character. The maximum length of the recognition sequence is 32 characters.

10. **Interrupt Keys**

There is a fundamental restriction of the core editor that interrupt keys must be a single keystroke (such as [Ctrl][G]), not a double keystroke (such as [Esc]-[F1]). This means that it is not possible to install keymaps files that have nested case statements in the Interrupt section. Note however that the only command that *must* be in the Interrupt section is Job.Interrupt. Everything else can be implemented as a non-interrupting sequence.

♦

HINT: Destroying a difficult to grab object

Applicable to: Delta 1	*Fix:*
References:	*Keywords:* Destruction
Originated from: Mv	*Revised by:* Pbk

Haven't you sometime had difficulties to get a good grip around the neck of a slippery object? This is the way to really poke your fingers all the way into the throat of that object:

1. Do a `Show_Directory_Information` on this object (`!Users.Sysmve.Temporary` in this example):

```
!USERS.SYSMVE.TEMPORARY
[DIRECTORY,149946,1] Information:
Name     : [DIRECTORY,149946,1] (TEMPORARY in [DIRECTORY,12573,1])
Library  : LIBRARY_CONTROL_POINT [DIRECTORY,12573,1] on Vpid 256
Class    : 1 (ADA); Subclass: 2 (WORLD); State: 7 (N/A); Control_Point
Switches: [DIRECTORY,138683,1]
Access   : Default: SYSMVE=>RW,NETWORK_PUBLIC=>R,SYSTEM=>RW
       Library: SYSMVE=>RCOD,NETWORK_PUBLIC=>R,MAILER=>R,SYSTEM=>RCO
Target   : <<Unknown>>
Versions: 1; Retention Count: 0; Default: [ADA,167064,1]
Version 1: [ADA,167064,1]   ( 1 in [DIRECTORY,149946,1])
No children.
```

   This command is located in `!Commands.System_Maintenance'Spec_View.Units`.

2. Note the object's internal *name* shown in the upper left-hand corner (right after `Name :`). In the example: `[DIRECTORY,149946,1]`.

3. Use that *name* surrounded by angle brackets to do the destruction:

```
Comp.Destroy (Unit => "<[DIRECTORY,149946,1]>",
              Threshold => 100,
              Limit => "<WORLDS>",
              Response => "<PROFILE>");
89/06/12 18:16:29 ::: [Compilation.Destroy ("<[DIRECTORY,149946,1]>", 100,
89/06/12 18:16:30 ... "<WORLDS>", PERSEVERE);].
89/06/12 18:16:30 +++ !USERS.SYSMVE.TEMPORARY has been marked.
89/06/12 18:16:30 --- The 1 marked object will now be destroyed.
89/06/12 18:16:30 +++  1 object has been destroyed.
89/06/12 18:16:31 ::: [End of Compilation.Destroy Command].
```

♦

BUG: Lib.Reformat_Image of archived unit corrupts the unit	
Applicable to: D1, D2	*Fix:* D3
References:	*Keywords:* Ada Unit, Corruption, Format
Originated from: Vnv	*Revised by:*

`Lib.Reformat_Image` of an Ada unit in the archived state corrupts the content of the unit. The original content cannot be restored.

Example:

```
procedure A_Silly_Program is
   Nothing : Integer;
begin
   null;
end A_Silly_Program;
```

Running `Lib.Reformat` on that unit and hitting then [Definition] gives the following in the file:

```
[error]
```

There is no workaround. `Lib.Reformat` should only be used on Ada units in the source state.
♦

BUG: Lib.Reformat_Image of archived unit corrupts the

INFO: What not to run Library.Compact_Library on	
Applicable to:	*Fix:*
References:	*Keywords:* Compaction, Group, Login, Password, Session
Originated from: Vnv	*Revised by:*

Do not compact the following libraries since they are cached by the system and strange things happen when you do.

- `!Machine.Users`
  Logins will fail with no passwords found.

- `!Machine.Groups`
  The commands in package operator which deal with groups will fail with a bad context (`name resolution`) problem.

- `!Users`
  Login will fail to find the user session and will not be able to create a session since the system can't find `!Users`.

These can be fixed by rebooting. Incidently `!Machine.Temporary` and `!Machine.Queues??` are also cached but the system reenters it into the cache if it is not there.

♦

TIP: Undeletable specifications due to nonexistent bodies	
Applicable to:	*Fix:*
References:	*Keywords:* Ada Unit, Body, Destruction
Originated from: Vnv	*Revised by:*

When trying to delete a specification which is in the `Source` state, that has no body or deleted body, you get the message:

```
OBSOLESCENCE_ERROR detected while deleting <Spec_Name>
Other objects depend on the selected unit.
```

You cannot create the body, it says it exists. If you try to `Lib.Move` it, you get the above message. If you copy the specification to another place, the problem is copied there too. There is but one solution:

1. Promote the specification to `Coded`.
2. Create an empty Ada unit and type in a body (this one cannot be promoted, it reports that a body already exists, even though none is listed in the directory).
3. In the enclosing directory, select the anonymous Ada object created in step 2, and `Lib.Rename` it to the name of the non-existent body. It works !
4. Then promote the new body to `Installed`.
5. You should now be able to demote and delete the body and the specification

Note: This has worked in one case where `Repair_Directory` has failed.
♦

PROBLEM: Session switches not properly initialized	
Applicable to: D2	*Fix:*
References:	*Keywords:* Code Generator, Command, Session, Switch
Originated from: Pbk	*Revised by:*

Command_Cg switches are not properly initialized for new sessions.

For a newly created session that does not have yet have a session switch file, the code generator supplies default values for session switches that are different from the default values contained in the session switch file once it is created. Since the code generator caches the values of these switches, the switch file and the code generator can disagree on the value of these switches and the switch editor appears to display the wrong value. To work-around the problem, you must change the value of any one of the Command_Cg session switches (and then change the value back, if desired). The affected switches appear to be:

```
Command_Cg.Retain_Delta1_Compatibility
Command_Cg.Check_Compatibility
Command_Cg.Page_Limit
Command_Cg.Full_Debugging
```

Note: If you create the session switch file before coding any units on that session, the problem does not occur.

♦

TOOL: A tool to change switches in several switch files	
Applicable to:	*Fix:*
References:	*Keywords:* Switch, Wildcard
Originated from: Rcp	*Revised by:*

The file name argument to `Switches.Set` must specify a single switch file, so the command:

```
Switches.Set ("R1000_Cg.Retain_Delta1_Compatibility := False",
              "!??'C(Switch)");
```

will not work to change this switch in all switch files. You may use the following procedure to do this:

```
with Directory;
with Log;
with Profile;
with Switches;
procedure Set_Switches (Switch_Values : String := ">> SWITCH SPEC <<";
                        Switch_Files : String := Switches.Associated;
                        Response : String := "<PROFILE> USE_ERROR") is

    function Is_Ok (Status : Directory.Naming.Name_Status) return Boolean is
        use Directory.Naming;
    begin
        return Status = Directory.Naming.Successful;
    end Is_Ok;

    procedure Report_Failed (Message : String) is
    begin
        Log.Put_Line (Message,
                      Kind => Profile.Negative_Msg,
                      Response => Profile.Value (Response));
    end Report_Failed;

begin
    declare
        Iterator : Directory.Naming.Iterator;
        Status : Directory.Naming.Name_Status;
    begin
        Directory.Naming.Resolve (Iterator, Switch_Files, Status);
        if not Is_Ok (Status) then
            Report_Failed ("Can't resolve """ & Switch_Files & """ because " &
                           Directory.Naming.Extended_Diagnosis
                              (Status, Switch_Files));
            return;
        end if;
        while not Directory.Naming.Done (Iterator) loop
            Switches.Set (Spec => Switch_Values,
                          File => Directory.Naming.Source_Name (Iterator),
                          Response => Response);
            Directory.Naming.Next (Iterator);
        end loop;
```

```
    end;
end Set_Switches;
pragma Main;
```

Invoking it with:

```
Set_Switches ("R1000_Cg.Retain_Deltal_Compatibility := False";
              "!??'C(Switch)");
```

will change the value of the `Retain_Deltal_Compatibility` switch in *all* switch files on the machine. `Set_Switches` should run fairly fast (definitely less than 1/2 hour).

Note: Changing *this particular switch* in some Rational-supplied worlds will cause them to fail.
♦

TIP: Locks on link packs

Applicable to:	*Fix:*
References:	*Keywords:* Link Pack, Lock
Originated from: Trw	*Revised by:*

It may happen that you have a remaining lock on a link pack (either in a world or in a subsystem). Every command attempting to modify the link pack yields:

```
Can't open link pack Illegal simultaneous access to link object was
attempted.
```

Even if everyone logs off (and there remain no background jobs), the lock remains. Doing `What.Locks` on the world shows no locks, because you have no way to designate the link pack.

The only way to view the lock is to use `Show_Locks` (located in `!Commands.System_Maintenance`):

```
Show_Locks (Show_Job_Action_Summary => True,
            Show_All_Actions => True,
            Job_Id_Filter => 0,
            Action_Id_Filter => 0,
            Show_Raw_Info => False);
```

This command dumps all active locks in the system, so it may be a good idea to have as few running jobs as possible.

In the output, you should look for locks on objects with manager class `12`, i.e. names of the form:

```
<[12, ..., 1]>
```

These are the link packs.

There should only be one; the output identifies the job that has the lock and the nature of the lock.
♦

PROBLEM: Cmvc.Initial fails with access error - ACLs seems OK	
Applicable to: may occur also in D2	*Fix:*
References:	*Keywords:* Acl, Cmvc
Originated from: Rjg	*Revised by:*

When trying to `Cmvc.Initial` in a world where you have RCOD access, it fails saying it needs RCOD access to enclosing world.

This can occur if the default ACL of the world is too restrictive. The reason for this is because `Cmvc.Initial` first creates the subsystem - this works OK because the enclosing world ACL is OK - but it creates a subsystem whose ACL is the default ACL of the enclosing world. The command then fails because it tries to create the working view within the subsystem.

Note that in D1, the failure looks completely different and even more cryptic, it fails with a `problem in Cmvc source storage` because it cannot create some file.
♦

PROBLEM: Cmvc.Initial fails with access error - ACLs

Rev. 0

PROBLEM: Create a view fails with Commit_Prevented exception	
Applicable to: D1	*Fix:* D2
References:	*Keywords:* Acl, Cmvc, Exception
Originated from: Rjg	*Revised by:*

This applies to any `Cmvc` commands that create views, such as `Cmvc.Make_Path` or `Cmvc.Release`. If it fails immediately with exception `!Implementation.Action.Commit_Prevented`, one thing to look for is the default ACL of the enclosing subsystem. You need RW default access to the subsystem in order to create new views. In D1, it fails with this non-specific error message, but is fixed in D2.

♦

PROBLEM: Cmvc.Make_Code_View raises Nonexistent_Page_Error	
Applicable to:	*Fix:*
References:	*Keywords:* Cmvc, Exception
Originated from: Mam	*Revised by:*

`Cmvc.Make_Code_View` must be run in a commands window separate from other `Cmvc` commands. There are occasions when the `Cmvc.Make_Code_View` will raise `Nonexistent_Page_Error` and not be able to create the code view. If this happens all you need do is reexecute the `Make_Code_View` procedure.

♦

PROBLEM: Cmvc.Make_Code_View raises

HINT: Subsystem lore	
Applicable to:	*Fix:*
References: CSR 3102 Subsystem, View	*Keywords:* Cmvc, Corruption, Design Facility, Destruction, Parent,
Originated from: Phl	*Revised by:* Rjg

This is a list of recipes and hints on hunting big game: subsystems. These *subsystem sporting guns* are likely to help you dealing with subsystems that refuse to be destroyed and repaired. Of course, all the data contained in these subsystems will be lost by the process.

Problem:

When running `Cmvc.Destroy_View`, you get:

```
*** Cannot open configuration <VIEW>.STATE.MY_CONFIGURATION due to a problem
... found by Cmvc Source Storage (Cannot open the Database).
*** If the above error is an internal error or is otherwise incomprehensible,
...  the structure of the view may be corrupt. Run Cmvc_Maintenance.
... Check_Consistency on the view and then use Cmvc.Destroy_View.
```

Solution:

In this case, running `Check_Consistency` doesn't fix the problem (it says everything is OK). You have to unfreeze and delete the file called `<View>.State.My_Configuration`, and then run `Check_Consistency` on the view.

Problem:

When running `Cmvc.Destroy_View` on a Design Facility view, you get:

```
*** Unable to update caches in Components.Withdraw (OPEN_ERROR, Bad view
... handle in Cache_Operations.Open (<VIEW>.UNITS.<UNIT> is not enclosed
... within a view.)).
```

Solution:

You simply have to go in the `<View>.Units` directory and demote `<Unit>` to source. Then retry to delete the whole view (you will probably have another error, since the view is likely to be corrupted).

Problem:

If you interrupt (`Job.Kill`) `Cmvc.Destroy_View`, or if it fails, you may well have a view that is of subclass `World`... but is still a view. This may or may not prevent deletion of the view. If it prevents deletion, try the following trick:

Solution:

You have to set the subclass to `Load_View` to be able to proceed. The command is:

```
Lib.Set_Subclass (Existing => "<SELECTION>",
```

```
To_Subclass => "Load_View",
Response => "<PROFILE>");
```

It may well happen that this command fails. Try it again with various subclasses appropriate for views: `Spec_View`, `Combined_View`, `System_View`, etc.

You should run `Check_Consistency` again before attempting to delete the view.

Note that setting the subclass to `Combined_View` seems to be nearly always possible, and doesn't raise further problems when checking consistency. Beware of `System_View`: in normal subsystems, it seems to create more problems than it solves.

Problem:

When running `Cmvc.Destroy_View`, you get the message:

```
*** Unable to create directory "<VIEW>_STATE" because the operation
... is illegal in this context.
*** If the above error is an internal error or is otherwise incomprehensible,
...  the structure of the view may be corrupt. Run Cmvc_Maintenance.
... Check_Consistency on the view and then use Cmvc.Destroy_View.
```

Solution:

You have to go in `<Subsystem>.Configuration`. There is a directory called `<View>_State` (hence the error when attempting to create it...). You delete this directory and the objects it contains, and then you may destroy the view.

Problem:

When deleting a view, you get the message:

```
*** Cannot open configuration <VIEW>.STATE.MY_CONFIGURATION due to a problem
... found by Cmvc Source Storage (The specifed object isn't a Configuration
... Object).
*** If the above error is an internal error or is otherwise incomprehensible,
...  the structure of the view may be corrupt. Run Cmvc_Maintenance.
... Check_Consistency on the view and then use Cmvc.Destroy_View.
```

In this case, `Check_Consistency` will say that it has fixed the problem with the configuration, but do not believe it, it lies: the problem may remain.

Solution:

Go to the `<Subsystem>.State` directory and delete the CMVC database. Then run `Check_Consistency` again on the whole subsystem.

Problem:

When running `Check_Consistency` on a view, you get the message:

```
*** Internal Error - Cmvc State Object 'state.imports' in view '<VIEW>'
... is missing.  Try running Cmvc_Maintenance.Check_Consistency to fix it.
```

So `Check_Consistency` asks you to run `Check_Consistency`... Isn't it lovely!

Solution:

Changing the view subclass (e.g. to `Combined_View`) is likely to allow `Check_Consistency` to run correctly.

Problem:

When running various `Cmvc` commands you get the message:

```
*** Cannot open view indicator file
```

Solution:

The view indicator is the `<Subsystem>.State.This_Is_The_Root_Of_A_View` file. It may be that this file has been deleted (in which case you have to re-create it) or that it bears too restricted ACLs (in which case they must be loosened).

Problem:

When running `Design.Set_Parent`, you get the following message:

```
*** <> is not a structural link object.
Parents is quitting after errors.
```

Solution:

`Design.Parent` and `Design.Children` give the same error. Running `Design.Check_Consistency` and `Cmvc_Maintenance.Check_Consistency` on the parent made the problem go away.

♦

PROBLEM: Constraint_Error while saving compatibility information	
Applicable to:	*Fix:*
References:	*Keywords:* Archive, Cdb, Cmvc, Exception, Subsystem
Originated from: Jgp, Rcp	*Revised by:*

Problem:
`Archive.Copy` of a primary subsystem to a secondary raises:

```
Exception Constraint_Error (Type Range) from PC blah while saving
compatibility info for <View>
```

and `Check_Consistency` and `Check.Views` report no problems.

Solution:
Most likely cause is having overly large CDB files. First place to look is:
`<Subsystem>.State.Compatibility.Declaration_Objects` and
`<Subsystem>.State.Compatibility.Offsets`

Do a `Common.Explain` on the whole directory and look at the size of the objects, and add up the total size of all objects. `Archive` chokes on copying CDBs larger than a certain size (Not sure if this is 2 Mb or 4 Mb).

Note: CMVC does not complain about the size of the objects, it will let you merrily create as large a CDB as your disk space allows. You will not encounter this problem until you try to `Archive.Copy` it.

Once you identify the large files, there are two methods of reducing their size:

Preferred Method

1. Run `Cmvc_Maintenance.Destroy_Cdb` on the subsystem (all secondaries as well as primary).

   Warning: This will demote all units in all views to source, and destroy all code views. If this is unacceptable to the customer, see the *Alternate Method* below.

2. Repromote all units in the primary and remake all code views.

3. `Archive.Copy` the primary CDB to all secondaries.

4. Repromote all units in all secondaries.

5. `Archive.Copy` the code views to all secondaries.

Alternate Method
This involves less demotion than the *Preferred Method* above, but must be done with extreme care. If any errors are made, they will have to use the *Preferred Method.*

1. Manually delete all code views in primary and all secondaries.

2. Manually demote the unit (spec and body) associated with each of the overly large objects in the `Declaration_Numbers` and `Offsets` directories (the ones you found above).

3. Manually delete the overly large objects in both the `Declaration_Numbers` and `Offsets` directories.

   Note: The object does not appear in the `Offsets` directory if the corresponding unit has never been installed. If the corresponding unit has been installed and the object does not appear in the `Offsets` directory, there are more serious problems with the CDB and you should consult with RCP instead of using either of these methods.

Then follow steps 2 through 5 under the *Preferred Method* for the units that were demoted.

If these methods do not work or the problem does not appear to be caused by having overly large CDB files, consult with RCP.

♦

PROBLEM: Policy_Error while restoring controlled units	
Applicable to:	*Fix:* D2
References:	*Keywords:* Archive, Cmvc, Controlled Object, Exception
Originated from: Jgp	*Revised by:*

There is a known problem when you try to restore a controlled unit using `Archive.Restore`. If the unit is not checked out, when you try to do the `Archive.Restore`, you will get:

```
90/01/05 16:40:36 *** You must check out <UNIT> to do that.
90/01/05 16:40:36 *** Can't recreate a version of <UNIT> (POLICY_ERROR).
90/01/05 16:40:36 ++* <UNIT> could not be restored.
```

To work around this, you must either make the units you are restoring uncontrolled, or check them out.

Note: If you are doing multiple restores to the same units, you will lose the history of these changes, unless you do a `Check_Out` before and a `Check_In` after each `Archive.Restore`.

♦

INFO: When a units content disagrees with that in the CMVC database

Applicable to:	*Fix:* after D2
References:	*Keywords:* Archive, Cmvc, Format, Switch
Originated from: Rjg	*Revised by:*

There are two known ways of inadvertently causing the following error message when `Cmvc_Maintenance.Check_Consistency` runs:

```
89/12/22 14:27:16 *** <UNIT>'s content disagrees with that in
89/12/22 14:27:16 ... the CMVC source database.  Use Check_Out,
89/12/22 14:27:16 ... Show_Image_Of_Generation, and Check_In as appropriate
89/12/22 14:27:16 ... to fix it.  If the object is supposed to represent the
89/12/22 14:27:16 ... last generation, and the content of the object is
89/12/22 14:27:16 ... correct, then simply checking it out and in will
89/12/22 14:27:16 ... correct the problem.
```

1. When the format switches in the library switches are changed, and a unit is deleted and restored or `Lib.Reformat_Image` is used. The fact that the format is different, even though it is syntactically identical, causes the inconsistency.

2. When a view is copied or restore using `Archive.Copy` or `Archive.Restore`, *and* any of the original Ada source units had more than one trailing `Ascii.Nl` character. `Archive` strips multiple trailing `Ascii.Nl`s, causing inconsistencies in the copied view.

In either case, the problem is not serious and can be ignored. If you must fix the problem, checking the unit(s) in and out can solve the problem. If it is a released view, you'll need to destroy the view and rebuild from the CMVC database. You may also delete the CMVC database and run `Cmvc_Maintenance.Check_Consistency`. At least one subsystem in the D1 software catalog, `Math_Library`, exhibits this problem.

♦

INFO: When a units content disagrees with that in the

TIP: Errors due to incompatibilities between spec and load views	
Applicable to:	*Fix:*
References: CSR 3268, CSR 3332 Subsystem, View, Visibility Error	*Keywords:* Compatibility, Exception, Operand Class Error,
Originated from: Ams, Cew	*Revised by:*

- `Visibility_Error` occurs at run-time due to an incompatibility between units in a spec and load view. The solution is to run `!Tools.Compatibility.Check.Views` and it'll tell you what units are incompatible. But what makes a unit incompatible?...

  This error has even been seen when the spec view was created from the load view via the `Cmvc.Make_Spec_View` procedure, and the unit that `Check.Views` flagged as incompatible have duplicate specs in both the load and spec view.

  Running `Check_Consistency` shows nothing unusual. `Check.Views` identifies the incompatible unit, but as I said, there is absolutely not difference between the two specs.

  This has been identified as a possible code generation bug, but as of now the only verified workaround is to copy the unit from the load view to a text file use `Compilation.Parse` to recreate the Ada object.

- The following error message has been seen when promoting a generic instantiation from Source to Coded. The error was caused by *with*ing different views of the same subsystem in the generic instantiation and in the generic specification.

  ```
  Long_Float_Type attribute Position_Component_Type'Last is not a value of
  type Long_Float_Type.
  ```

- There is one documented case where a load/spec view incompatibity between a view instantiating a generic and a view outside the closure of the instantiating unit raises `!Lrm.System.Operand_Class_Error`. This is likely to occur in other cases as well.

♦

TIP: Errors due to incompatibilities between spec and

Rev. 2

PROBLEM: CMVC source control max line length	
Applicable to:	*Fix:* Epsilon
References:	*Keywords:* Cmvc, Controlled Object, Maximum
Originated from: Rjg	*Revised by:*

This max length is approximately 1000 characters. It depends somewhat on the number of leading blanks, so it is not exact. This is a hardcoded value, and will stay that way until Epsilon, at which time it will be unlimited.

♦

TIP: Cannot create configuration on Cmvc.Make_Path

Applicable to:	*Fix:*
References:	*Keywords:* Configuration, Joined Objects, Reservation Token, Subsystem, View
Originated from: Rjg	*Revised by:*

When running:

```
Cmvc.Make_Path (New_Path_Name => "<Pathname>",
                Join_Paths => False); -- Not the default.
```

you get the message:

```
Cannot create configuration <Pathname>_Working due to a problem in Cmvc
source storage (specified reservation token name is in use)
```

This may happen when you try to create a view with the same name as a previously destroyed view.

Here are two things to try:

1. Look for `<Subsystem>.Configurations.<Pathname>_Working` and `<Subsystem>.Configurations.<Pathname>_Working_State`. If they exist, do a `Lib.Destroy` on them, then do a `Cmvc_Maintenance.Expunge_Database`. The problem was probably caused when a view was destroyed without specifying `Destroy_Configurations_Also => True` as a parameter.

2. If the above named configuration objects do not exist, then the old view was probably destroyed properly. It is possible that the old view had some object(s) joined with another view, causing the old reservation token names to remain in use after the destruction of the old view. `Cmvc.Make_Path` picks a specific name based on the view name - this causes the conflict. Fortunately, `Cmvc.Sever` picks an unused name, so you can get the desired effect by doing the following:

   ```
   Cmvc.Make_Path (New_Path_Name => "<Pathname>",
                   Join_Path => True);
   Cmvc.Sever (What_Objects => "?");
   ```

◆

PROBLEM: Problems with shared ancestors in Cmvc.Sever	
Applicable to:	*Fix:*
References:	*Keywords:* Joined Objects, Subsystem, View
Originated from: Rjg	*Revised by:*

When running `Cmvc.Merge` on two working views, you may end up with a message like this:

```
%%% Internal Error - Unhandled exception
   !Implementation.Cmvc_Implementation.Unknown_Error (PC= #5EE00B, #24D)
```

This has been observed in the following situation: three joined views were created, the *original* was severed and destroyed, then a `Cmvc.Merge` was attempted on the remaining two. Apparently `Cmvc.Sever` does not destroy all dependencies on common ancestors. Thus the two views being merged referenced the nonexistent view, causing the problem.

♦

PROBLEM: Problems with shared ancestors in

PROBLEM: Cmvc.Initial fails when the model world contains a switch file

Applicable to:	*Fix:*
References:	*Keywords:* Cmvc, Model, Subsystem, Switch
Originated from: Pbk	*Revised by:*

1. Create a model world that contains a library switch file. Problem shows up when switch file is *not* associated with the enclosing model world. This can best be done by doing `Lib.Rename` on the switch file after it is associated with the enclosing world.

2. Use `Cmvc.Initial` with defaults except for the name of the system object and the model (specify the above created model). The subsytem will be created, but will fail when creating the working view. The message is:

   ```
   *** Unable to get dir object for switch file because operation is
   illegal in this context
   ```

   This error message is much too cryptic!

◆

PROBLEM: Cmvc.Initial fails when the model world

Rev. 0

TOOL: Deleting view objects	
Applicable to:	*Fix:*
References:	*Keywords:* Cmvc, Destruction, Subsystem, View
Originated from: Tom	*Revised by:*

Use this procedure on objects that can't be deleted by `Cmvc.Destroy_View` or `Lib.Delete`. After running this, `Lib.Delete` will work.

```
with Directory;
with Io;
procedure Reset_Controlled (Name : String := "<SELECTION>") is

    package Dir renames Directory;
    package Naming renames Dir.Naming;
    package Object_Ops renames Dir.Object_Operations;
    Iter : Naming.Iterator;
    Nstatus : Naming.Name_Status;
    Obj : Directory.Object;
    Status : Directory.Error_Status;

    Abandon : exception;

    function "=" (X, Y : Naming.Name_Status) return Boolean
        renames Naming."=";
    function "=" (X, Y : Dir.Error_Status) return Boolean renames Dir."=";
    function Image (Obj : Directory.Object) return String
        renames Naming.Get_Full_Name;

    procedure Check (Status : Dir.Error_Status; Error_Message : String) is
    begin
        if Status /= Dir.Successful then
            Io.Put_Line (Error_Message & " (" &
                         Dir.Error_Status'Image (Status) & ")");
            raise Abandon;
        end if;
    end Check;

    procedure Check (Status : Naming.Name_Status; Error_Message : String) is
    begin
        if Status /= Naming.Successful then
            Io.Put_Line (Error_Message & " (" &
                         Naming.Name_Status'Image (Status) & ")");
            raise Abandon;
        end if;
    end Check;

begin
    Naming.Resolve (Iter, Name, Nstatus);
    Check (Nstatus, "Couldn't resolve the name");

    while not Naming.Done (Iter) loop
```

```
        Naming.Get_Object (Iter, Obj, Status);
        Check (Status, "Couldn't get object from iterator");

        Object_Ops.Set_Controlled (Obj, False, Status);
        Check (Status, "Couldn't reset the controlled bit for " &
               Image (Obj));

        Object_Ops.Set_Slushy (Obj, False, Status);
        Check (Status, "Couldn't reset the slushy bit for " & Image (Obj));

        Io.Put_Line ("+++ Reset bits for " & Image (Obj));

        Naming.Next (Iter);
    end loop;
exception
    when Abandon =>
        null;
end Reset_Controlled;
```

◆

HINT: Programmatically determining if a view is a release or code view		
Applicable to:	*Fix:*	
References:	*Keywords:* Directory, View	
Originated from: Dhe, Jls	*Revised by:*	

The question is: given a `Directory_Tools.Object_Handle` or a `Directory.Object` which is known to denote a load view, how to determine if it is a code view or release view. It would be nice to have a subclass to interrogate, but we do not have that yet.

There are two solutions:

1. Code views have an object called `Code_Database` inside the view, released views do not. If the object exists, it is a code view, if it does not exist, it is a regular load view.

2. Use `Object_Info.Any.Is_Code_Only_Load_View` from the Software Catalog. If you haven't yet looked at the spec for `Object_Info`, please do so. This toolbox has every widget you need to ask questions about objects in the Environment.

♦

HINT: Programmatically determining if a view is a

INFO: Relocation logging	
Applicable to:	*Fix:*
References:	*Keywords:* Cmvc, Mirror, Relocation
Originated from: Tom	*Revised by:*

Some information about Cmvc errors can be gained by turning on relocation debugging.

1. In !Implementation create the following specification:

```
package Relocation is

    pragma Subsystem (Directory);
    pragma Module_Name (4, 3916);

    pragma Consume_Offset (21);

    procedure Enable_Debugging (Sense : Boolean := True);

end Relocation;
```

2. Execute Relocation.Enable_Debugging. This causes a log to be created in !Machine.Temporary for most Cmvc operations that are executed on the machine.

3. Run the Cmvc command that was causing the problem

4. Look in !Machine.Temporary for a file with the name:

   Relocation_Debug_Log_<User_Name>_<Session_Name>_<Job_Number>

   This file may contain useful information about the error.

5. When done, run Relocation.Enable_Debugging (False).

◆

TIP: Cmvc_Hierarchy.Build_Activity gets old views	
Applicable to:	*Fix:*
References:	*Keywords:* Activity, Subsystem, System, View
Originated from: Rjg	*Revised by:*

`Cmvc_Hierarchy.Build_Activity` by default builds an activity that includes the *latest* views of child subsystems. It uses the last check-in time of `<View>.State.Release_History` to determine which view is the latest. Certain commands, such as `Cmvc.Check_Consistency`, `Cmvc.Check_Out` and `Cmvc.Check_In`, modify the date of this object. In some instances this can cause the wrong view (in particular, spec view) to be referenced in the system activity created by `Build_Activity`. The simple fix is to manually check-out and check-in `Release_History` in the view you want to be considered *latest*.

◆

PROBLEM: Problem with Program.Current and code views	
Applicable to:	*Fix:*
References:	*Keywords:* Activity, View
Originated from: Rjg	*Revised by:*

```
function Current (Subsystem : String := "<SUBSYSTEM>";
                  Unit : String := "<PROCEDURE>";
                  Parameters : String := "";
                  Activity : String := "<ACTIVITY>") return String;
-- Constructs a procedure call suitable for Run or Run_Job that references
-- the appropriate view...
```

Unfortunately, if the given activity and subsystem specify a spec/code view pair, this function returns an invalid path name. It returns <CODE_VIEW>.Units.<PROCEDURE>, which does not exist - there are no executable units in code views, only in load views. The units in code views can only be referenced via the spec view.

This problem has the interesting result that a program may work when run against spec/load views, but fail when run against spec/code views.

♦

PROBLEM: Problem with Program.Current and code

Rev. 0

PROBLEM: Incorrect Work_Order_Implementation shipped with D2	
Applicable to: D_12_1_1	*Fix:*
References:	*Keywords:* Work Order
Originated from: Doug, Jgp	*Revised by:*

It has been determined that some functionality is missing from package `!Implementation.Work_Order_Implementation` as shipped in D_12_1_1. Attempting to use this package without correcting this situation will cause `Operand_Class_Errors`.

The following is an excerpt from the output produced by `File_Utilities.Differences`. File #1 is how it should be, file #2 is how it is. In order to avoid future confusion, it is recommended to apply these changes at the earliest convenience.

First difference, around line 535:

```
   procedure Create_Info (The_Order :     Work_Order_Handle;
                          The_Info  : out User_Info;
                          Success   : out Status);

   procedure Close_Info (The_Order :     Work_Order_Handle;
                         The_Info  : out User_Info;
                         Success   : out Status);
1  --
1  function Create_User_Name (The_Order : Work_Order_Handle) return String;
1
1  function Close_User_Name (The_Order : Work_Order_Handle) return String;

   ------------------------------------------------------------

   function Get_Notes (For_Order : Work_Order_Handle) return String;
   --
   -- Constraint_Error will be raised if the handle is not open.

   procedure Set_Notes (To_Value  :     String;
                        For_Order :     Work_Order_Handle;
                        Success   : out Status);
```

Second difference, around line 610:

```
   procedure Count_Comments (For_Work_Order :     Work_Order_Handle;
                             Comments       : out Natural;
                             Success        : out Status);

   generic
       with procedure Visit (The_User    : User_Id;
1                            User_Name   : String;
                             The_Comment : String;
                             The_Element : Element_Name;
                             The_Time    : Calendar.Time);
   procedure Traverse_Comments
```

```
                (For_Work_Order : Work_Order_Handle; Success : out Status);
   --
   -- Do not Add_Comments while traversing.
   -- Comments are time-ordered
1  -- The User_Name may be good even if the The_User no longer
1  -- exists.  If the comment was generated in a Delta 3 system
1  -- then the user name is stored separately.
2
   end Work_Order_Operations;
```

And the third difference, around line 900:

```
  end Work_Order_List_Display;

  ------------------------------------------------------------------------
  --======================================================================--
  --======================================================================--
  ------------------------------------------------------------------------
1
1  package Controlled_Operations is
1
1       -- These operations are not meant to be used lightly, as
1       -- they can cause objects to end in an inconsistent state.
1       -- They are intended for maintenance and archival purposes only.
1       -- If you don't know exactly what they do, you shouldn't be using
1       -- them.
1
1       procedure Set_Create_Info (The_Order : Work_Order_Handle;
1                                  The_Info  : Work_Order_Operations.User_Info;
1                                  Success   : out Status);
1
1       procedure Set_Close_Info (The_Order : Work_Order_Handle;
1                                 The_Info  : Work_Order_Operations.User_Info;
1                                 Success   : out Status);
1
1       procedure Set_Status
1                    (The_Order  : Work_Order_Handle;
1                     The_Status : Work_Order_Operations.Status_Enumeration;
1                     Success    : out Status);
1
1       procedure Set_Default (The_Field : in out Work_Order_Field;
1                              Value     :        String;
1                              Success   : out    Status);
1       procedure Set_Default (The_Field : in out Work_Order_Field;
1                              Value     :        Boolean;
1                              Success   : out    Status);
1       procedure Set_Default (The_Field : in out Work_Order_Field;
1                              Value     :        Integer;
1                              Success   : out    Status);
1
1       procedure Add_Order          (The_Venture :     Venture_Handle;
1                                     The_Order   :     Work_Order_Id;
1                                     Success     : out Status);
1       procedure Add_Default_Order (The_Venture :     Venture_Handle;
1                                    The_User    :     User_Id;
```

```
1                                   The_Order    :     Work_Order_Id;
1                                   Success      : out Status);
1       procedure Add_List         (The_Venture  :     Venture_Handle;
1                                   The_List     :     Work_Order_List_Id;
1                                   Success      : out Status);
1       procedure Add_Default_List (The_Venture  :     Venture_Handle;
1                                   The_User     :     User_Id;
1                                   The_List     :     Work_Order_List_Id;
1                                   Success      : out Status);
1
1   end Controlled_Operations;

   private
```

A fix tape has been prepared by JGP and widely distributed. Unfortunately, it does only work if all dependents of `Work_Order_Implementation` are coded before installing the fix. If this is not the case, then, immediately after restoring the tape, you should change lines 456 to 459 of file:

```
!Machine.Release.Archive.Environment.D_12_1_1.Command_Data.Steps
```

to add option `Promote` in the `Archive.Restore`:

```
Archive.Restore (Objects =>
                    "!Implementation.Work_Order_Implementation'Spec",
                 Options => "Replace, Promote", --<<HERE>>
                 Device => "Work_Order_Implementation");
```

◆

PROBLEM: Incorrect Work_Order_Implementation

Rev. 0

BUG: Default generic parameter raises Illegal_Reference	
Applicable to: all releases	*Fix:*
References: LRM 12.3.6	*Keywords:* Compiler, Exception, Generic Parameter, Illegal Reference
Originated from: Pbk	*Revised by:*

Instantiation of a generic package, using defaults subprogram generic parameters obtained from another instantiation, blows at elaboration time with exception `System.Illegal_Reference`.

```
with Outer_Gen;
procedure Illegal_Ref is
   package Outer is new Outer_Gen (Integer);
   procedure Inner is
      new Outer.Inner_Gen; --<< Outer.Def for default; blows there
begin
   Inner;
end Illegal_Ref;

generic
   type Tt is (<>);
package Outer_Gen is
   procedure Def (X : Tt);
   generic
      with procedure P (X : Tt) is Def;  --<< default to Def
   procedure Inner_Gen;
end Outer_Gen;

with Io;
package body Outer_Gen is
   procedure Def (X : Tt) is
   begin
      Io.Put_Line ("Def executed");
   end Def;
   procedure Inner_Gen is
   begin
      P (Tt'First);
   end Inner_Gen;
end Outer_Gen;
```

Work-around: Do not use default; name subprogram explicitly:

```
package Outer is new Outer_Gen (Integer);
procedure Inner is new Outer.Inner_Gen (P => Outer.Def);
```

♦

PROBLEM: Native compiler rejects subtype of private type	
Applicable to: D1	*Fix:* D2
References: LRM 7.4.1(4)	*Keywords:* Compiler, Private Type
Originated from: Phl	*Revised by:* Jls, Pbk

Native compiler rejects subtypes of a private type while coding the unit, with some *internal error.*

Consider the following declarative part:

```
type T (D : Positive) is
    record
        null;
    end record;
type R (D : Integer) is
    record
        S : T (D);
    end record;
subtype Sr is R (-1);   -- Constraint_Error raised here !
```

There is no ambiguity as where the exception should be raised, but if you consider this:

```
package Bug is
type R (D : Integer) is private;
subtype Sr is R (-1);
-- more declarations here
private
type T (D : Positive) is
    record
        null;
    end record;
type R (D : Integer) is
    record
        S : T (D);
    end record;
end Bug;
```

It is not clear where the exception should be raised: on the subtype declaration ? well the type R is not elaborated yet !! [LRM 7.2(3)]. To make it more obvious, let us refine this example:

```
package Bug is
type R (D : Integer) is private;
subtype Sr is R (-1);

private
Bound : Integer := Get_A_Number;
subtype Rng is Integer range Bound .. Integer'Last;
type T (D : Rng) is
    record
        null;
    end record;
type R (D : Integer) is
```

```
    record
        S : T (D);
    end record;
end Bug;
```

The RM does not help much in deciding where the exception should be raised, but AI-0007 provides a clear decision on how to handle this situation.

Work-around: Declare subtypes in another package...

This problem is fixed in D2, provided the `Retain_Deltal_Compatibility` switch is set to `False`.

♦

BUG: Code generator error causes Stack_Frame_Errors	
Applicable to: D1, D2, D3	*Fix:* D4
References:	*Keywords:* Code Generator
Originated from: Rcp, Rjg	*Revised by:*

Code generator errors (`Stack_Frame_Error`) occur when coding a subprogram which passes an array slice to another subprogram, when the array is inside a discriminated record which has a variant part and the length of the array is defined by a discriminant (whew!).

Simplest test case fails with `Stack_Frame_Error` on D_10_20_0 and D_11_1_3:

```
procedure Cg_Bug2_Test is
   type Variants is (First, Second);
   subtype String_Length is Positive range 1 .. 10;
   type Fatal_Record (Which_Variant : Variants; Length : String_Length) is
      record
         Fatal_Array : String (1 .. Length);
         case Which_Variant is
            when First =>
                Dummy : Integer;
            when Second =>
                null;
         end case;
      end record;

   procedure Call_Me_With_A_Slice_If_You_Dare (Foo : in out String) is
   begin
      null;
   end Call_Me_With_A_Slice_If_You_Dare;

   procedure Cant_Code_This_Subprogram (Rec : in out Fatal_Record)
      is separate;
begin
   null;
end Cg_Bug2_Test;

separate (Cg_Bug2_Test)
procedure Cant_Code_This_Subprogram (Rec : in out Fatal_Record) is
begin
   Call_Me_With_A_Slice_If_You_Dare (Rec.Fatal_Array (1 .. 5));
end Cant_Code_This_Subprogram;
```

Work-around: Declare an array constant to hold the slice, as in:

```
procedure Cant_Code_This_Subprogram (Rec : in out Fatal_Record) is
begin
   declare
      Dummy : constant String := Rec.Fatal_Array (1 .. 5);
   begin
      Call_Me_With_A_Slice_If_You_Dare (Dummy);
   end;
```

```
end Cant_Code_This_Subprogram;
```

◆

INFO: Machine_Restriction and other strange errors	
Applicable to:	*Fix:*
References:	*Keywords:* Exception, Restriction
Originated from: Rcp, Swb, Vnv	*Revised by:*

If you get something like `Machine_Restriction` while running an Ada program on the R1000 the developpers (SWB, RCP) may be able to figure out why! This describes how to collect the information they need.

1. Run code under the debugger until the exception is raised, and then execute in a command window:

```
Debug.Enable (Variable => Debug.Addresses, On => True);
Debug.Information (Info_Type => Debug.Exceptions, For_Task => "");
```

   This will display:

```
Task ROOT_TASK, #335467:
   Exception Numeric_Error (overflow) raised at: .RAISE_EXCEPTION.1d.
   Control offset: #16,  Raise Pc: #6F903, #1D, Exception name: (#21)
```

2. Get the `Raise Pc` code segment id and instruction eg. `#6F903, #1D`.

3. Subtract 8 (Hex 8) from the Instruction eg. `#1D - 8 = 15`.

4. Execute:

```
Debug.Memory_Display
   (Address => "#6F903, #15", Count => 2 * 8, Format => "CODE");
Debug.Enable (Variable => Debug.Addresses, On => False);
```

   This will display:

```
Memory_Display ("#6F903, #15", 16, CODE);
#6F903, #15: #0000  Action, Illegal
#6F903, #16: #0000  Action, Illegal
#6F903, #17: #0000  Action, Illegal
#6F903, #18: #001E
#6F903, #19: #0004
#6F903, #1A: #0002
#6F903, #1B: #6004  Indirect_Literal, Discrete #0020
#6F903, #1C: #00E2  Load_Encached, Value_02
#6F903, #1D: #03EC  Declare_Variable, Discrete, With_Value,
                    With_Constraint
#6F903, #1E: #4501  Exit_Subprogram 1
#6F903, #1F: #0000  Action, Illegal
#6F903, #20: #0000  Action, Illegal
#6F903, #21: #0000  Action, Illegal
#6F903, #22: #8000  Call 0, 0
#6F903, #23: #0000  Action, Illegal
#6F903, #24: #0000  Action, Illegal
```

5. Send this with the pertinent code to developers (CG group).

◆

PROBLEM: Pragma Private_Eyes_Only and generics	
Applicable to:	*Fix:*
References:	*Keywords:* Generic, Pragma
Originated from: Trw	*Revised by:*

Closed private parts cannot be implemented for generic packages because of R1000 architectural restrictions. (Although SOME cases can be implemented, they are very difficult to characterize and so, from the start, we designed the system to not allow closed private parts for any generics.) Basically, when the R1000 declares a type (at execution time), it must know the details of the type or have an existing type on hand from which to get the details. Except in the case of generics the code generator is either able to find the information, by looking through private parts, or can find a type that it can use as a model.

As an example of a specific case that does not work, consider the following:

```
generic
package Gen is
  type T is private;
private
  type T is ...
end Gen;

with Gen;
package Exam is
  type C is private;
  type Ac is array (...) of C;
    ...
  package Inst is new Gen;
    ...
private
  type C is new Ac.T;
end Exam;
```

The problem is the declaration of `C`. It is at the `Type C is private` declaration that the R1000 needs the information about the type. The code generator looks forward to the full type declaration and follows derivations, etc, until is finds the details of the type or a reference to a type that will exist at run-time when the first part of the type declaration is attempted. (If `Inst` were a local package **not an instantiation** the code generator would look into its private part for the completion. Nested package specs aren't *really* closed.) In this case, since the private part of the generic would be closed, there would be no compile-time information, hence the only option left is to use an existing type to serve as the model, but the only model is the private type of the generic, which at the first part of the declaration of `C` doesn't even exist! And that's the problem.

If the declaration of `C` could be moved after the instantiation there would be no problem, but the declaration of `Ac` prevents that; or the architecture could allow declaration of `C` with incomplete infomation about the generic's private type, but that all take more work.

♦

BUG: Unable to exit named loop in select statement	
Applicable to: D1,D2	*Fix:* D3
References: LRM 5.7 (3)	*Keywords:* Loop, Semanticize
Originated from: Gbd, Hjl	*Revised by:*

It is impossible to semanticize this unit. One error is marked, on the loop name A_Loop in the exit statement. The error message is: A_LOOP does not denote a loop [LRM 5.7 (3)]

```
procedure Ta16 is

  task A_Task is
     entry An_Entry;
  end A_Task;

  task body A_Task is
  begin
     select
        accept An_Entry;
     else
        A_Loop:
            loop
                exit A_Loop;
            end loop A_Loop;
     end select;
  end A_Task;

begin
    null;
end Ta16;
```

Work-around: A block statement A_BLOCK surrounding A_LOOP will allow D_10_20_0 to semanticize this unit.

```
    select
        accept An_Entry;
    else
        begin
            A_Loop:
                loop
                    exit A_Loop;
                end loop A_Loop;
        end;
    end select;
```

All named loops in the else part of a select statement will behave badly in D2 and earlier. This should be been fixed in D3.

♦

INFO: Storage_Error (allocation) while using access types	
Applicable to:	*Fix:*
References:	*Keywords:* Access Type, Allocation, Representation Clause
Originated from: Mam	*Revised by:*

When you declare an access type a pointer of 24 bits is used. This means that a collection of 2**24 will be allocated for that type. If the type is too big to fit in that space then the above message will result when a object of that type is created. To solve this one, simply increase the size of the pointer. The maximum size is 32 bits. If you use 32 bits you will be using up all of the modules data space. That means that no more objects can be declared in the given module. A module is package or a task.

Example:

```
package A is
    S : Constant Integer := 2**24;
    type Chars is array (1 .. S) of Character;

    --- type B is too big to fit in a standard collection.
    type B is
        record
            A1 : Chars;
            A2 : Chars;
        end record;
    type C is access B;
end A;


with Io;
package body A is
    F : C := new B;
begin
    Io.Put_Line ("This is part of elaboration");
end A;


with A;
procedure B is
begin
    null;
end B;
```

When you run B you get, in the message window:

```
B'BODY'V(1) terminated due to unhandled exception Storage_Error (allocation)
```

You can simply fix this by adding the representation clause for `C'Size` after the access type declaration. We use 28 in this case because it's enough:

```
    ...
    type C is access B;
```

INFO: Storage_Error (allocation) while using access


```
for C'Size use 28;
...
```

♦

INFO: Tasks in variant records		
Applicable to: D_10_20_0, D_11_1_0		*Fix:* D3
References:	*Keywords:* Code Generator, Exception, Internal Error, Record Type, Task	
Originated from: Rcp	*Revised by:*	

The message:

```
Promote failed - Code generation errors found
1: INTERNAL_ERROR Unhandled exception in code generation Constraint_Error (
Variant), from PC = #3E8014, #18E8
```

was received when attempting to code the following program:

```
package Task_Package is
    task type Task_Type is
    end Task_Type;
end Task_Package;

with Task_Package;
package Client_Package is
type Client_Type (Value : Boolean) is
    record
        case Value is
            when True =>
                X : Task_Package.Task_Type;
            when False =>
                null;
        end case;
    end record;
end Client_Package;
```

The problem appears to be that the code generator cannot handle code generation for a variant record type when one of the variants contains a field of a task type. The possible workarounds are to:

1. Move the task field from the variant part to the fixed part.

2. Somehow turn the task field into a non-task field ... e.g., introduce an access type; turn the task type into a truely private type (i.e., turn it into a private type exported by a package with a closed private part).

◆

INFO: Compatibility between D1- and D2-style code objects	
Applicable to: D_11_3_@ and later	*Fix:*
References:	*Keywords:* Code Generator, Code View, Compatibility, Loaded Procedure, Switch, View
Originated from: Mam, Swb	*Revised by:*

- Moving from a D2 to a D1 machine

  Code objects such as loaded procedure and code views cannot be moved directly from a D2 to a D1 machine.

  If you need to create a loaded procedure or a code view on a D2 machine which will be moved to a D1 machine you need to set the compatibility switch in the world where the code object will be created. The switch is called `R1000_Cg.Delta1_Code_View_Compatibility`. For example if you are creating a Loaded Procedure in `!Local` from `!Users.Operator.Main_Program` you must set the library switch in `!Local` before running `Compilation.load`.

  To create a code view set the switch in the load view which is to be released as a code view. When the code view is created this switch file will be copied. When the code objects are created they will be in D1 format.

  Don't forget to set the switch back again.

- Taking advantage of the new D2 features

  The proper way to do this to set the switch `R1000_Cg.Retain_Delta1_Compatibility` to `False` in all the worlds you are interested in, uncode all the units in those worlds (i.e. demote them to Installed), and recode them. Uncoding and recoding is the bulk of the work.

  You don't have to do this all at once if you don't want to: old-style units (i.e. units coded with D1-compatibility enabled) can reference new-style units and vice versa.

  The main rule to keep in mind here is that corresponding units in a spec view and a load view must be coded with the same switch setting or a link-time error will result. Note that recoding a spec view with a new switch setting will make any existing code views useless, so you probably want to destroy these views and regenerate them; there are similar issues with loaded procedures.

◆

TIP: Too many instructions in code segment		
Applicable to:	*Fix:*	
References:	*Keywords:* Code Generator, Code Segment, Compiler, Instruction, Maximum	
Originated from: Gbd, Rcp	*Revised by:*	

There is a size limitation imposed by the R1000 native code generator. When running `Common.Promote` on an Ada compilation unit that will require too many R1000 instructions to be generated, the `Semantic error(s) found` message appears but no other indication of what is wrong. When `Compilation.Promote` or `Compilation.Make` are run, the output window contains messages like the following:

```
--- Messages generated while promoting <Unit> to CODED.
---   33184  instructions for package <Unit>
*** Assembly Error : Too many instructions in code segment,
...  Program too Large.
++* CODE_GENERATION_ERROR returned while attempting to
... promote <Unit> to CODED.
```

The native code generator allows a maximum of 32,768 R1000 instructions per compilation unit. The limit cannot be changed. Changing switches `Default_Job_Page_Limit` or `R1000_Cg.Page_Limit` will not help. The only way to solve this problem is to move enough declarations or statements to other compilation units so that the number of instructions is less than 32,768. The number of instructions mentioned in the messages generated by the compiler is the number of instructions currently required for the compilation unit.

Note that the limit of 32,768 instructions applies per Ada *compilation unit*, not per *library unit*. Because of this, you can reduce the number of instructions in a library unit package body by creating body stubs for some subprograms and moving their code to subunits.

♦

PROBLEM: System.Machine_Restriction raised during compilation	
Applicable to: d_10_20_0a	*Fix:*
References:	*Keywords:* Compiler, Exception, Machine Restriction
Originated from: Djd	*Revised by:*

If you try to compile a program that is using an unitialized variable as an index of a string, for instance:

```
A : Integer;
B : String (1 .. A);
```

you will get the message:

```
unhandled exception !Lrm.System.Machine_Restriction
```

◆

PROBLEM: System.Machine_Restriction raised during

Rev. 0

PROBLEM: Tree compaction error	
Applicable to:	*Fix:*
References:	*Keywords:* Compiler, Internal Error, Promote, Tree
Originated from: Unknown	*Revised by:*

When trying to install units in a spec view, you may have the following message:

```
*** Internal error tree compaction error
*** other error returned while attempting to promote <UNIT>
```

In this case, demoting all units in the spec view to `Archived` and re-installing everything fixes the problem.

♦

TIP: Fixing Ada units with Set_Unit_State		
Applicable to:	*Fix:*	
References:	*Keywords:* Ada Unit, Demote, Exception, Mirror, State	
Originated from: Mam	*Revised by:*	

While trying to demote an Ada unit, you get a message like:

```
%%% Demotion failed with !LRM.System.Nonexistent_Page_Error,
    from PC=#FD4C16, #909
++* OTHER_ERROR returned while demoting <UNIT>
```

You may use the following mirror to change the unit's state:

```
with Directory;
with Action;
package Directory_Utilities is

    pragma Subsystem (Directory, Closed);
    pragma Module_Name (4, 3951);
    pragma Consume_Offset (3);

    procedure Set_Unit_State (The_Version : Directory.Version;
                              The_State : Directory.Declaration_Operations.
                                          Declaration_State;
                              The_Status : out Directory.Error_Status;
                              The_Action : Action.Id := Action.Null_Id;
                              Max_Wait : Duration := Directory.Default_Wait);

end Directory_Utilities;
```

♦

TIP: Removing incrementally-added circular withs	
Applicable to: D1	*Fix:* D2
References:	*Keywords:* Ada Unit, Demote, With Clause
Originated from: Jls	*Revised by:*

In D1, it is possible to incrementaly a *with* in such a way that two specifications can form a mutual dependency, which is *not* permitted by the RM. In so doing, a chicken-and-egg problem is created, in which neither spec can be demoted to remove the circularity because each needs the other to be demoted *first*.

For example:

```
with B;
package A is
end A;

with A;
package B is
end B;
```

Work-around:

Type the following into a library:

```
pragma Switch (Subsystem_Interface);
package Tree_Repair is
    pragma Module_Name (4, 3917);
    pragma Consume_Offset (10);
    procedure Remove_Item (Item : String := "<SELECTION>");
end Tree_Repair;
```

Now select one of the offending *with* clauses, create a command window off the unit with the selection, and execute this command:

```
Tree_Repair.Remove_Item;
```

This removes the *with* clause from both the DIANA tree and the image, and does so *without checks*. The image goes off the screen and when you revisit the unit the *with* is gone. Now you may demote both units.

Obviously, this is *very dangerous*, and should only be done to fix this one problem and then the `Tree_Repair` package *must* be deleted from the machine. Use with *extreme caution*.

In D2, it is no longer possible to incrementally add circular *withs*.

♦

HINT: Removing DIANA attribute Read_Only	
Applicable to: All releases (?)	*Fix:*
References:	*Keywords:* Destruction, Mirror, Read-Only
Originated from:	*Revised by:* Mv

An Ada unit can be marked internally as *read only*, by a DIANA attribute.

One way to achieve this is with `pragma Read_Only` which should not be used. (A user can accidentally get this by copying and coding one of the environment supplied packages such as `!Lrm.System`). This has the effect that the object can not be deleted nor demoted. Message is:

```
1: ERROR <package name> cannot be demoted because it is read-only.
```

The *read only* attribute can be overridden using the following mirror:

```
package Tree_Repair is
    pragma Module_Name (4, 3917);
    pragma Consume_Offset (3);
    procedure Remove_Read_Only (From_Unit     : String  := "<SELECTION>";
                                Remove_Pragma : Boolean := False);
end Tree_Repair;
```

The boolean parameter `Remove_Pragma` defines whether the attribute should be taken away from the DIANA tree or not. For destruction purposes, the setting of this parameter doesn't matter. Successful activation gives a message such as:

```
+++ Removed read_only attribute from <Unit>
```

♦

BUG: Coding of subunit fails with unhandled exception	
Applicable to: seen in D1	*Fix:*
References:	*Keywords:* Compiler, Diana, Exception, Promote, Subunit
Originated from: Rjg	*Revised by:*

```
promote failed with unhandled exception <UNIT=4818533,ORD=1>,
     from PC=#51AC14,#128DC
```

This could be caused by a screwed up DIANA representation. It would be possible to copy the units to another directory, and coding would still fail. Currently it is unknown what might cause the DIANA to go bad, but certain kinds of editing in a declarative region that contains subunit declarations may be responsible, but this is a somewhat wild guess. The problem was observed in a generic package that declares a complex private type (discriminated record containing array of records, some components are generic formals) where the subunits (ten of them) have this complex type as a parameter. Please add or clarify this writeup if you encounter this problem and have more information.

How to fix it:

As with past problems with screwed-up DIANA, you must force the DIANA to be regenerated. You can either:

- Demote the offending unit to `Archived` and repromote.
- Or select the entire image of the offending unit delete it with `Editor.Region.Delete` and finally `Editor.Hold_Stack.Top` (better known as [Region]-[↑]).

After either of these operations, the subunits should code successfully.

Note: In the observed case, the *offending unit* was the package body where the subunits were declared, not the subunits themselves.

♦

PROBLEM: Cannot promote a semantically correct Ada unit		
Applicable to:	*Fix:*	
References:	*Keywords:* Diana, Exception, Garbage, Promote	
Originated from: Vnv	*Revised by:*	

Problem Part 1:

User is working on a 1000 lines Ada unit semanticizing regularly and comes to a point where semanticize does NOT return errors or the `No semantic errors` message. If the user then tries to [Promote] the code he gets:

```
Unhandled exception: storage_error (allocation) from PC mumble mumble
```

Solution Part 1:

What has happened is that the DIANA garbage pail has become full and needs emptying. Two actions can do this: a successful [Promote], or a

```
Compilation.Demote(goal => Compilation.Archived)
```

.

Note: Sometimes the user/session error log will contain useful extra info that the user didn't get in their message window. To save this error log it must be copied to a file before the user logs back into that session, or it will be lost.

Problem Part 2:

Sometimes when the above solution is tried the user may get the following when trying to re-promote the unit:

```
Unhandled exception: !Lrm.System.Illegal_Heap_Access
```

Solution Part 2:

If you cannot login to the system (with a developer) then try the following:

- Do a [Definition] of the file in question
- `Write_file` the unit to a file of your choice
- `Delete` the original file.
- Do a `Library.Expunge` using the defaults on the library that contained the now deleted original file.
- Do a `Compilation.Parse` of the text file your created with `Write_File`

Normally Solution Part 1 should clear up such problems without having to resort to Solution Part 2.
♦

PROBLEM: Cannot promote a semantically correct Ada

Rev. 0

PROBLEM: Debugger completed when starting the debugger	
Applicable to:	*Fix:*
References:	*Keywords:* Activity, Debugger
Originated from: Vnv	*Revised by:*

When activating the debugger, you get a message `Debugger completed` before it even starts.

Check your activity for line near beginning that has <...>:

```
Subsystem      | Spec View        |    Load View       | Context
-------------------------------------------------------------------------
<...>
CI             | REV9_1_SPEC      |    CODE9_1_0       | !TOOLS
COMPATIBILITY  | REV9_1_SPEC      |    CODE9_1_0       | !TOOLS
DEBUGGERS      |                  |    CODE9_1_6       | !MACHINE.RELEASE
```

Select it, delete it, promote the activity, it should work then... all else being OK.

◆

PROBLEM: Debugger completed when starting the

Rev. 0

BUG: Session iterator gets a nonexistent session	
Applicable to:	*Fix:* D2
References:	*Keywords:* Iterator, Non-Existent, Session
Originated from: Rjg	*Revised by:*

Depending on how a session is logged off, this symptom can occur (usually by force logoff): one of the sessions on the iterator is an old logged out session.

The right way to determine if a session is logged out is to compare `System_Utilities.Last_Logout` with `System_Utilities.Last_Login`: if `System_Utilities.Last_Logout > System_Utilities.Last_Login`, then it is a logged out session.

Note: `System_Utilities.Logged_in` will do you no good. It will say that the *phantom* session is logged in !

♦

TIP: Servers that won't start at boot time	
Applicable to:	*Fix:*
References:	*Keywords:* Design Facility, Exception, Initialize, Server
Originated from: Vnv	*Revised by:*

Some of us have experienced a server that starts fine if we initiate it from a command window, but will not start if we put it into `!Machine.Initialize`.
The problem may be that the program is trying to write a message to the message window. But there is no message window available when the system boots so it would display:

```
!!! unhandled exception !Io.Io_Exceptions.Status_Error (not open)
```

in the error logs. As an example lets look at `!Machine.Initialize_Design_Facilities`

```
    Command_Options : constant String :=
       "Input | Output | Error => !MACHINE.DEVICES.NIL";
     ...
begin
    ...
    Program.Create_Job (Registration_Command,
                        Context => Get_Context (Registration_Command),
                        Options => Command_Options,  <-- NOTE
                        Job => Job_Id,
                        Status => Status);
```

Putting this in the `Program.Create_Job` command cured the problem just fine.

This problem has reared itself a couple times in slightly different ways. Another variation is that the program runs fine when you run it in the debugger, but when you try to run it as a server in `Initialize`, it bombs out. Many environment commands (like the print spooler) will write things to the message window. If you call them in a server you may get these results.
◆

PROBLEM: Terminal hangs after logoff		
Applicable to:	*Fix:* D2	
References:	*Keywords:* Fa Queue, Hang-Up, Quit, Session, Terminal	
Originated from: Djl, Pam	*Revised by:*	

After issuing the `Quit` command, the screen goes blank, but nothing else ever happens, and the terminal will not respond to any input, power cycle, reset, etc.

Doing a `What.Users` shows that the users are still logged on (long after issuing the `Quit` commands) and are trying to execute the `Quit` command.

This turns out to be a bug in the microcode where a job has terminated but is left in `FA_QUEUE` state. This job is using some resource needed by the `Quit` command. Since the job is terminated (in a strange manner), it cannot be killed, and the `Quit` jobs cannot complete. The system continues to run just fine and users can save their work. They just can't log off. Unfortunately the only recourse in this situation is to reboot. This bug is fixed in Delta 2, and it is unlikely that this will happen again after rebooting.

How do you tell if you're in this sorry state? From EEDB:

```
Kernel: show_task_states
CACHE/DISK [CACHE]: [CR]
```

If you see any messages with `IN_FA_QUEUE` like the following:

```
16#7A8D4#; IN_FA_QUEUE maybe for ( 212,MODULE, 427); <blah blah>
```

then try another `Show_Task_States`. It is possible to catch a normal job in the `IN_FA_QUEUE` state, but it should only be in that state for a few microseconds. If the second try shows the same message with the same job number (first number in parentheses--in this example, `212`), you are probably in this locked state. To verify this, do:

```
Kernel: job_name
VPID [4]: <job number, 212 in above example>
```

If the name of the job (last column of table) is `<?>`, the job is terminated and stuck `IN_FA_QUEUE`. In this case, you should track down DJL to look at the problem.

In some cases it just takes a long time to execute the `Quit` procedure.

♦

TOOL: Job killer server	
Applicable to:	*Fix:*
References:	*Keywords:* Job, Kill, Server
Originated from:	*Revised by:*

The following program is a server job which prompts the operator's console for a job number to kill. If you answer it, it will kill the job if it isn't a system job and is either attached or detached. As the job killer is a server, it won't be stopped by a system reaching `Stop_Jobs`.

```
with Io;
with Job;
with Machine;
with Directory;
with Scheduler;
with Disk_Daemon;
with System_Utilities;
with String_Utilities;
procedure Job_Killer is
   I, O : Io.File_Type;
   Console : constant String := "!Machine.Devices.Terminal_1";
   Prompt : constant String := "Job Name To Kill: ";
   type User_Job is new Natural range 6 .. 255;
   Job_Numbers : constant String := User_Job'Image (User_Job'First) & " .. " &
                                    User_Job'Image (User_Job'Last);
   procedure Do_Kill (Job_Id : Machine.Job_Id;
                      Session_Id : System_Utilities.Session_Id;
                      Job_Image : String;
                      Job_Name : String;
                      User_Name : String;
                      User_Session : String) is
      Kind : Scheduler.Job_Kind := Scheduler.Get_Job_Kind (Job_Id);
      The_Session_Object : Directory.Object :=
         System_Utilities.Session (Session_Id);
      The_Session : constant String :=
         Directory.Naming.Get_Full_Name (The_Session_Object);
   begin
      if String_Utilities.Equal (User_Name, "*system") then
          Io.Put_Line (O, "Can't kill a " & User_Name & " job");
      else
          case Kind is
              when Scheduler.Attached | Scheduler.Detached =>
                  Io.Put_Line (O, "Killing user job" & Job_Image &
                                  ", " & User_Name & ", " &
                                  The_Session & ", " & Job_Name);
                  Job.Kill (Job_Id, The_Session);
                  Job.Enable (Job_Id, The_Session);
              when others =>
                  Io.Put_Line (O,
                               "Can't kill a " &
                                  Scheduler.Job_Kind'Image (Kind) & " job.");
          end case;
```

```
        end if;
    end Do_Kill;
    procedure Do_Killing (J : String) is
        Job_Number : User_Job;
        Job_Id : Machine.Job_Id;
    begin
        Job_Number := User_Job'Value (J);
        Job_Id := Machine.Job_Id (Job_Number);
        if Job_Number not in User_Job then
            raise Constraint_Error;
        end if;
        Do_Kill (Job_Id, System_Utilities.Get_Session (Job_Id),
                 Machine.Job_Id'Image (Job_Id),
                 System_Utilities.Job_Name (Job_Id),
                 System_Utilities.User_Name
                    (System_Utilities.Get_Session (Job_Id)),
                 System_Utilities.Session_Name
                    (System_Utilities.Get_Session (Job_Id)));
    exception
        when Constraint_Error =>
            Io.Put_Line (O, J & " is not a job number in the range " &
                            Job_Numbers);
        when others =>
            Io.Put_Line (O, Machine.Job_Id'Image (Job_Id) &
                            " isn't an active job.");
    end Do_Killing;
begin
    Scheduler.Set_Job_Attribute (System_Utilities.Get_Job, "Kind", "Server");
    Disk_Daemon.Set_Prevent_Stop_By_Warning (True);
    Io.Open (O, Io.Out_File, Console);
    Io.Open (I, Io.In_File, Console);
    loop
        Io.Put (O, Prompt);
        Do_Killing (Io.Get_Line (I));
    end loop;
    Io.Close (I);
    Io.Close (O);
end Job_Killer;
```

♦

BUG: System hangs, users cannot login		
Applicable to: D1	*Fix:* D2 (may be)	
References:	*Keywords:* Deadlock, Hang-Up, Job, Login, Session, Terminal	
Originated from: Cbh, Mam, Rjg	*Revised by:*	

When a user tries to log in, she/he can get past *username*, *password*, and *session* prompts; then she/he gets the `last login at ...` message. At this point the session hangs (no further output). The following error message is displayed on the system console:

```
!!! Session_Switches_Probable_Semaphore_Deadlock; _Meta_Mutex_Is_Held;
Idle; Wait_10
```

There are various scenarios that might cause this.

1. The first one is when a user rapidly logs in and logs out. The problem exists in D1. A bug that causes the problem has been found and fixed for D2, however there may be other bugs that cause the same problem. This problem is expected to be seen very rarely. The only cure is to reboot the system.

2. The second one is when a job is disabled (by `Job.Disable` or by the Kernel command `Disable`), and it holds the `Meta_Mutex` semaphore. Try re-enabling all disabled job; it may fix the problem. If so, you will have, on the system console, the message:

   ```
   Session_Switch_No_Deadlock_After_All; _Reader_#3A0E1; Wait_10
   ```

   indicating that the semaphore has been released.

Theory:

`Meta_Mutex` is the highest level semaphore in the operating system. If some job grabs it and doesn't release it, nothing else can run. This problem has been seen when users log in and out multiple times quickly, in which case rebooting is the only solution. In the second case, when the user disabled her/his jobs, one of them ended up with the semaphore. By enabling the job, the semaphore was released, and the system was "unhung".

◆

INFO: Job priorities	
Applicable to:	*Fix:*
References:	*Keywords:* Job, Priority
Originated from: Marlin	*Revised by:*

The table below shows the three kinds of priorities:

- Real machine priorities (the R1000 column),
- Ada priorities (`pragma Priority`),
- MTS priorities (or job priorities).

Notice that for R1000 priorities, best is the lowest number; for the other two, best is the highest number. The machine always executes the task with the best machine priority which is ready to run. Therefore, a task at machine priority 4 will execute only when all tasks at machine priorities 0..3 are blocked; runnable tasks within the same machine priority are round-robin scheduled.

```
      R1000    Ada    MTS
     ====================
Best     0
         1
         2
         3
        -----------------
         4       5
         5       4
         6       3
         7       2
         8       1
         9       0      6      MTS Foreground
        -----------------
        10       5
        11       4
        12       3
        13       2
        14       1
Worst   15       0      0      MTS Background
```

The user can only manipulate the Ada and MTS priorities. The Ada priority is set exactly as stated in the LRM: by a `pragma priority`. It can only be set on a task or main program, and is fixed for that task, but is subject to change during execution of a rendezvous (consult your LRM for the precise rules). The range is defined by type `System.Priority`. Since the pragma is part of the Ada program, it can only be changed by recompiling. The default is priority 1 (not 2, as stated in the LRM appendix).

The MTS priorities are what the documentation identifies as job priorities. There are really only two values possible, 6 for foreground jobs, and 0 for all background jobs. The values in between are not used. A foreground job has MTS priority 6; a background job has MTS priority 0. The MTS priority is a base priority for every task in the job. The Ada priority for each task is added to the MTS base priority for the

job to determine the real machine priority at which that task will execute. When a job detaches and becomes a background job, the Ada priorities within the job remain the same, but are mapped onto a different set of machine priorities.

A user cannot directly modify the MTS priority of a job once it has started or as part of initiating that job. The priority depends on whether the job is running as a foreground job (attached) or as a background job (detached, e.g. by hitting [Ctrl]-[G], or was started by `Program.Run_Job`).

`Pragma priority` sets only the Ada priority of a task within a job. The values are specified by the `System.Priority` type.

Package `Job` only indirectly allows changing the priority of a job via `Job.Connect` or `Job.Disconnect`. Disconnecting a job changes the job from foreground to background (hit [Ctrl]-[G]); connecting restores it to foreground (run `Job.Connect`). Accordingly, the MTS priority for the job changes along with the disconnect or connect.

`Scheduler.Get_Cpu_Priority` returns the MTS, or job, priority of a job; it returns the same value as `System_Utilities.Priority`.

Also, the manual may be a bit vague about the operations for setting job priorities because it has been thought of allowing more than 0 and 6 for MTS, i.e. job, priorities. There are currently no plans to do so, however.

♦

INFO: How to kill servers		
Applicable to:	*Fix:*	
References: CSR 3658	*Keywords:* Archive, Ftp, Job, Kill, Queue, Server, Session	
Originated from: Ams, Phl, Rjg	*Revised by:*	

- Jobs running under `Network_Public`.

  Certain jobs, such as the Print Spooler and the Archive Server, are started under `Network_Public`. These can be killed with `Job.Kill`, specifying `!Machine.Network_Public_Session` as the session:

```
Job.Kill (The_Job => 191,
          The_Session => "!Machine.Network_Public_Session");
```

- Print Spooler

  The Print Spooler can also be killed by `Queue.Kill_Print_Spooler`.

- FTP servers

  When you do an `Ftp.Connect`, the receiving machine spawns a job under the user and session names provided in parameters `Username` and `Account`. On the receiving machine, `What.Users` shows something like:

```
PHL                 -  218  IDLE      5.627  Ftp Server Session 8 phl
```

  Note that the session name is not displayed. If however you know it (e.g. `!Users.Phl.S_2`), you can kill this server by doing:

```
Job.Kill (The_Job => 218,
          The_Session => "!Users.Phl.S_2");
```

- General case

  You may kill *any* job (except those running under `*SYSTEM`) by doing:

```
Job.Kill (The_Job => >>ID<<,
          The_Session =>
             Directory.Naming.Get_Full_Name
                    (Directory.Object'
                        (System_Utilities.Session
                            (System_Utilities.Get_Session ( >>ID<< )))));
```

  Make sure that you provide the same job id for both parameters, otherwise the job will not be killed.

◆

TIP: How to unlock an object	
Applicable to:	*Fix:*
References:	*Keywords:* Action, Editor, Lock, Mailbox
Originated from: Phl	*Revised by:*

It may happen that an object remains locked, and you cannot kill the job that has created the lock. This is for instance the case if, for some reason, an Editor remains active with a lock on a mailbox.

To unlock the object, you must first find out what action has created the lock by typing:

```
Action_Utilities.Lock_Information (Name => "!Users.Anne.Mailbox.@");
```

This brings you a listing analogous to the one you get with `What.Locks`:

```
!USERS.ANNE.MAILBOX.FILE_NAME'V(2)
      No locks.
!USERS.ANNE.MAILBOX.MAIN_HEADERS'V(3)
      Updater: Action id = 1354898, Owner = 16#46A848#, ANNE.S_1 Job 72
!USERS.ANNE.MAILBOX.MAIN_SET'V(3)
      No locks.
```

Then, you must terminate this action, using the following program:

```
with Action, Io, Unchecked_Conversion;
procedure Finish_Action (Action_Id : in Integer) is
    function Uc is new Unchecked_Conversion
                          (Source => Integer, Target => Action.Id);
    The_Id : Action.Id := Uc (Action_Id);
begin
    Action.Finish (The_Action => The_Id,
                   Do_Commit => False,
                   Do_Inform => True);
    Io.Echo_Line ("Action" & Integer'Image (Action_Id) &
                  " successfully closed");
exception
    when others =>
        Io.Echo_Line ("Action" & Integer'Image (Action_Id) &
                      " could not be closed");
end Finish_Action;
```

You should call this program with the action id you obtained from `Lock_Information`. In this example:

```
Finish_Action (1354898);
```

♦

PROBLEM: Too large jobs hang in wait state	
Applicable to:	*Fix:* After D2
References:	*Keywords:* Hang-Up, Job, Working Set
Originated from: Rjg	*Revised by:*

When the working set size of a job is set very high (around 20 Kb), it is possible for the job to hang in a wait state. A 20 Kb working set size is very close to the maximum possible, because other parts of the Environment use the rest. When the job reaches the limit (this is unusual), it goes into wait state. It needs about 10% more to get out of wait state, which is above the maximum possible - so it never gets out.

There are two ways to fix this problem:

1. Lower working set size (16 Kb should be OK).

2. When the problem occurs, disable memory scheduling, then reenable after job has restarted

♦

BUG: Search_List editor crashing the system		
Applicable to: D1 and D2	*Fix:*	
References: PRS #9723630-0101-4	*Keywords:* Crash, Editor, Search List	
Originated from: Seh	*Revised by:* Lhrp, Rjg	

While using the `Search_List` editor it is possible to crash the system due the conjunction of a microcode error and an editor flaw:

- Invoke the search list editor with `Sledit`
- Insert a component name with [Object]-[I]
- The component is specified as <REGION>, e.g. `Add(Component => "<REGION>"...)`
- Select a region containing a valid library
- Press [Promote]; the <REGION> token is inserted in the Search list (not its value)
- Select this entry and then delete it
- Wait for a while (1 sec - 3 min)

While doing this you can observe the following on the console terminal:

```
!!! IMAGE_OBJECT_String_AT UNHANDLED_EXCEPTION in task 16#C2CB3#
```

- In Delta 1, the machine crashes due to a processor microcode error.
- In Delta 2, the microcode bug has been fixed, which means the machine will no longer crash as before. The software problem is *not* fixed, so it is still possible to insert <REGION> into your searchlist. When you attempt to delete it, you will get the same console message as before:

  ```
  !!! IMAGE_OBJECT_String_AT UNHANDLED_EXCEPTION in task ...
  ```

  In fact, from this point forward you will get this message for every command you attempt to execute (even `Quit`). You now need to `Op.Force_Logoff` the hosed session from another session. When you log back on to same session, it will behave normally except the <REGION> will continue to appear in the searchlist.

  To get rid of the <REGION>, *don't* use [Object]-[D] or `Search_List.Delete`. You may choose one of the following solutions:

  1. Delete the entire search list, located in `!Machine.Search_Lists`. You will continue to get the old search list even after deleting it until you log out and back in. The next time you log in with the desired session you will get the machine's default search list.
  2. Copy (`Region.Copy`) the search list editor display into a text file, delete the line with <REGION>, and use `Search_list.Revert` to copy the text file back into your search list.

♦

TIP: Search Lists changed after a machine boot	
Applicable to:	*Fix:*
References: CSR 3716	*Keywords:* Boot, Corruption, Search List
Originated from: Trw	*Revised by:*

Problem:

Users noted that their search lists had changed after a machine boot. The default search was modified on a Wednesday, the machine was rebooted on the following Friday, and about 90% of the users' search lists were modified. There *may* been a library deleted which was at one time well used but now obsolete, i.e. it would have been in most search lists.

Theory:

When a user logs in, his search list file, `!Machine.Search_Lists.<User>_<Session>`, is cached for faster access. In this process an attempt is made to resolve all library names (without wildcards or special symbols) in his search list file. If *any* of these resolutions fail, his search list file is ignored and the system default search list, `!Machine.Search_List.Default` is cached instead. If that is bad, a hard-wired search list is cached instead.

Only users whose search list referenced the deleted directory are affected. Of course, if all search lists were a copy of the default then it would seem they would all be affected. Also, by default, a new user shares the default search list with others until he changes it, which is another way users could get the impression that all search lists were affected.

Note that in all of this process, *no permanent search list file is changed.* Restoring the directory would fix the problem for everyone. Or, if everyone is using the default search list, fixing just it would fix the problem for everyone. Users would have to log off and log back on to get the new search list file cached for their session.

The current action of throwing the entire search list out because of one bad entry is a bit drastic. It's written this way to ensure that anyone who logs in has visibility to enough of the system to get around if his or the default search list gets damaged: you're realy hosed if you don't get anything on your search list.

◆

PROBLEM: Exception Device_Error due to long log messages	
Applicable to:	*Fix:*
References:	*Keywords:* Device Error, Error Log, Exception
Originated from: Hjl	*Revised by:*

If, in the middle of a log, you get a message like:

```
%%% internal error, unhandled exception !Io.Io_Exceptions.Device_Error,
from PC = #C1B40B, #112D
```

it may be that the length of the message that is being logged is too large. This may for instance happen with the sign-on line for a command which echos the names of many units being operated on.

To prevent it you either have to not print the log message (sign-on messages are auxiliary messages and can be filtered) or decrease the size of the message, by running the commands on smaller sets of files.
♦

PROBLEM: Exception Device_Error due to long log

Rev. 0

INFO: Raw I/O to a non window device		
Applicable to:	*Fix:*	
References:	*Keywords:* Input Output, Rs232, Terminal	
Originated from: Djd	*Revised by:*	

It seems like a reasonable thing to want to hook up a terminal or other RS232 device to one of the R1000's ports and talk to it with some kind of *raw* io. Unfortunately, `Window_Io.Raw` only supports I/O to a Rational window that is on an enabled terminal and connected to a session.

Our documentation is a little thin on how to do raw I/O to another RS232 device. The trick is to use `Device_Independent_Io.Open` and put some special options in the `Form` parameter. The defaults are:

```
Form => "CRLF = True, Echo = True, Editing = Line"
```

To do raw I/O you should use:

```
Form => "CRLF = False, Echo = False, Editing = False"
```

Also, do *not* make a mistake in the syntax of the `Form` parameter (like forget to use a comma) because `Open` will take it, not complain, and just use the defaults.

♦

HINT: Remote print queue server	
Applicable to:	*Fix:*
References:	*Keywords:* Printing, Queue, Remote, Server
Originated from: Mv	*Revised by:* Mv

To start up a server that enables printing on a printer physically attached to a remote R1000 host using the network.

The server is started on the local host from which the print job is to be submitted. It is recommended to include the code in one of the initialize files residing in `!Machine` to enable automatic start up of the server at boot time.

The remote print queue is specified as:

```
"!!" & <remote host name> & "!" & <class name>
```

Example:

```
Queue.Print (Name => "<IMAGE>", Options => "CLASS => !!MOUSE!LASER");
```

to access print queue `Laser` on remote host **Mouse**.

```
begin
   Program.Run_Job
      (Program.Current ("!tools.rpc_servers", "queue_service.start",
                        Activity => "!machine.release.current.activity"),
       Context => "!machine.error_logs",
       Options => "output = !machine.error_logs.queue_service_output," &
                  "error = !machine.error_logs.queue_service_error," &
                  "user = network_public, password = ()");
exception
   when others =>
       Error_Reporting.Report_Error
          (Caller => "!Machine.Initialize_Servers.Queue_Service.Start",
           Reason => Error_Reporting.Create_Condition_Name
                        ("Unhandled_Exception", Error_Reporting.Problem),
           Explanation => Debug_Tools.Get_Exception_Name (True, True));
end;
```

♦

INFO: Laser printing on A4 size paper		
Applicable to:	*Fix:*	
References: PostScript Reference Manual		*Keywords:* A4, Postscript, Printing, Queue
Originated from: Mv	*Revised by:*	

The print spooler was not designed with European size paper (A4) in mind. However, for printing plain text on a laser printer, one can adjust the print to A4 size rather well.

In `!Machine.Queues.Queue` there is a text file named `Plain_Text_Prefix` (if it doesn't exist, it will automatically be created when the print spooler is started).

At the end of `Plain_Text_Prefix`, locate the section beginning with `/BeginPage` and ending with `/EndPage {showpage State restore} def`. Use the example below to locate the lines to be replaced; those lines are *italicized* in the example replace them with the new lines, those are **bold** in the example) Example:

```
/BeginPage
{/State save def font setfont
%    [{0.5 INCH 10.5 INCH translate}
 [{0.55 INCH 10.65 INCH translate 0.95 dup scale}
%    {90 rotate 0.5 INCH -0.5 INCH translate}
 {90 rotate 1.1 INCH -0.6 INCH translate 0.95 dup scale}
%    {90 rotate 0.33 INCH -0.833 INCH translate 2 3 div dup scale}
 {90 rotate 0.43 INCH -0.75 INCH translate 0.7 dup scale}
%    {1.25 INCH 16.125 INCH translate 2 3 div dup scale}] MD get exec
 {0.6 INCH 11.3 INCH translate 0.7 dup scale}] MD get exec
} def
/EndPage {showpage State restore} def
```

The % is a comment prefix in PostScript.

♦

INFO: Configuring the print spooler for a PostScript printer		
Applicable to: D_10_20_0	*Fix:*	
References: SMU, System Manager's Guide		*Keywords:* Postscript, Printing, Queue
Originated from: Mv	*Revised by:* Mv	

This is an example on how to configure the print spooler for a PostScript laser printer connected to terminal port 29, using software flow control.
If necessary, restart the print spooler.

```
Queue.Restart_Print_Spooler;
```

Add the device. The `Laser_Comm` option is the key to success when configuring the spooler for a PostScript printer.

```
Queue.Add (Device => "TERMINAL_29", Options => "XON_XOFF, LASER_COMM");
```

Create the class.

```
Queue.Create (Class => "LASER");
```

Register the device to the class.

```
Queue.Register (Device => "TERMINAL_29", Class => "LASER");
```

If preferred, make the class the default class.

```
Queue.Default (Class => "LASER");
```

Enable the device for printing.

```
Queue.Enable (Device => "TERMINAL_29");
```

Example on how to configure the terminal port. This should be added to `!Machine.Initialize_Terminals'Body`.

```
Terminal.Set_Flow_Control (Line => 29, To_Be => "XON_XOFF");
Terminal.Set_Receive_Flow_Control (Line => 29, To_Be => "XON_XOFF");
Terminal.Set_Log_Failed_Logins (Line => 29, Enabled => False);
Terminal.Set_Disconnect_On_Failed_Login (Line => 29, Enabled => False);
Terminal.Set_Disconnect_On_Logoff (Line => 29, Enabled => False);
Terminal.Set_Logoff_On_Disconnect (Line => 29, Enabled => False);
```

♦

TIP: Killing an active print job	
Applicable to:	*Fix:*
References:	*Keywords:* Destruction, Printing, Queue
Originated from: Deg	*Revised by:*

It can be done, contrary to belief. And it will not corrupt the other jobs in the queue.

1. `Queue.Kill_Print_Spooler`
2. Find your file in `!Machine.Queues.Print_Spooler.Temp`
3. Delete it with `Common.Object.Delete` ([Object]-[D])
4. `Queue.Restart_Print_Spooler`

For non active jobs use `Queue.Cancel`.

♦

BUG: Not possible to queue to logical devices	
Applicable to: D1 at least	*Fix:*
References:	*Keywords:* Printing, Queue
Originated from: Deg	*Revised by:*

The procedure described in the SMU manual (Page 92) about using a logical device does not work. The print spooler uses `Terminal_Specific` to write to the device, which will fail for *non terminal* devices and thus automatically disable the device.

♦

INFO: Maximum number of devices registered to a class	
Applicable to:	*Fix:*
References:	*Keywords:* Printing, Queue
Originated from: Deg	*Revised by:*

There is a limit of 6 devices that can be registered to a class. There is no limit to the number of classes that can be registered with a device.

♦

INFO: Connecting a printer to a terminal server	
Applicable to:	*Fix:*
References:	*Keywords:* Network, Printing, Queue, Tcp Ip, Telnet, Terminal Server
Originated from: Phl	*Revised by:*

This card explains how to connect a printer to a RS-232 port of a terminal server, itself connected to a TCP/IP network, and how to configure the Print Spooler on a R1000 so that it correctly access this printer.

Terminal servers are generally DCEs, and printers DTEs. This means that "AM" and "AF" connectors should be used to connect the printer to the server. Similarly, "AM" and "AF" connectors should be used to connect terminals to the server. Choose a port to be devoted to the printer, and connect the printer to this port.

You will have to check and possibly modify the parameters of the printer, the terminal server, and the Print Spooler. Generally, modifying the parameters of the terminal server require to switch to *administrative mode*, which is often protected by a passsword. You should obtain this password, a copy of the documentation of the terminal server, and a copy of the documentation of the printer before starting the installation.

The communication parameters of the printer should be identical to those of the port of the terminal server to which it is connected. These parameters are: number of bits per character, number of stop bits, parity, receive speed, transmit speed, flow control. You should make these two sets of parameters identical, but their value is irrelevant to the R1000. You should adjust the speed to be the highest possible value for the printer.

For preliminary testing, you may also replace the printer by a terminal. In this case too, the communication parameters of the terminal and the port to which it is connected should be identical.

Now the most difficult problem is to explain to the Print Spooler how it may reach this printer. There is no standard way to address one of the RS-232 ports of a terminal server, but two strategies are commonly used by vendors:

1. Assign a different IP address to each port of the server. The Print Spooler may reach the printer by connecting to this address.

2. Assign only one IP address to the server, and a different socket id to each port. The Print Spooler may reach the printer by connecting to the specified socket of the server's address.

Both IP addresses and socket ids are TCP/IP concepts. Refer to a documentation concerning TCP/IP for more information about these.

Now, the Print Spooler can be configured to print through any TCP/IP server, by adding a Telnet port as a server, for example:

```
Queue.Add (Device => "Terminal_255", Options => "Host => The_Server, Socket =>
(NN,PP)");
```

The `Device` parameter must be a Telnet port name, in the range `Terminal_224` to `Terminal_255`.

You may choose whatever Telnet you like, this is not visible outside the R1000. The `Host` option is a name that resolves (via package `Transport_Name`, i.e. via `!Machine.Tranport_Name_Map`) to the network (e.g. `TCP/IP`) and host id (e.g. `89.2.3.4`) to which the Print Spooler is to connect when it has something to print. The `Socket` options is the socket number to which to connect. If omitted, it defaults to `(0,23)`.

These options are sufficient to deal with the two strategies above; other strategies may exist, but it is not sure that the Print Spooler will be able to handle them.

1. In the first case, you add entries in `!Machine.Transport_Name_Map` for each port of the server. For instance:

```
tcp/ip 89.1.0.1 server_port_1  -- the printer will be there
tcp/ip 89.1.0.2 server_port_2
tcp/ip 89.1.0.3 server_port_3
tcp/ip 89.1.0.4 server_port_4
```

   and you tell the Print Spooler to use port #1 for printing:

```
Queue.Add(Device => "Terminal_255", Options => "Host => Server_Port_1");
```

2. In the second case, you add one entry in `!Machine.Transport_Name_Map` for the server itself. For instance:

```
tcp/ip 89.1.0.1 server
```

   Then you have to determine what socket id is to be used to reach the printer. If the documentation of the server is unclear, you may try the following program. It connects to every socket id (from `(0,23)` to `(127,23))` and send a short message to this socket. The right socket is the one that produces the message `xmit OK`.

```
with Io, System, Transport, Transport_Defs, Transport_Name;
procedure Test_Server_Port
          (Host : in String := "";
           Text : in String := "This is a test" & Ascii.Cr & Ascii.Lf) is
    use Transport, Transport_Defs, Transport_Name;
    Address : constant Host_Id := Host_To_Host_Id (Host);
    C : Connection_Id;
    S : Status_Code;
    Cnt : Natural;
    Str : System.Byte_String (Text'Range);
begin
    for I in Str'Range loop
        Str (I) := Character'Pos (Text (I));
    end loop;
    for I in 0 .. 127 loop
        Open (C, S, "TCP/IP");
        Io.Put_Line ("open socket" & Integer'Image (I) & ' ' & Image (S));
        Connect (C, S, Address, (System.Byte (I), 23), Max_Wait => 10.0);
        Io.Put_Line ("connect socket" & Integer'Image (I) &
                     ' ' & Image (S));
        Transmit (C, S, Str, Cnt, 10.0);
        if Cnt = Str'Length then
            Io.Put_Line ("xmit OK");
        else
```

```
            Io.Put_Line ("xmitted only" & Integer'Image (Cnt) & " bytes");
        end if;
    end loop;
end Test_Server_Port;
```

Once you have determined the right socket, you tell the Print Spooler to use it for printing:

```
Queue.Add(Device => "Terminal_255",
          Options => "Host => Server, Socket => (NN,23)");
```

♦

INFO: Spooler device error

Applicable to:	*Fix:*
References:	*Keywords:* Device Error, Printing, Queue, Tcp Ip, Telnet
Originated from: Smp	*Revised by:*

If you are trying to print to a network device, and get the following message in your message window:

```
from System: Spooler device error; request 192 will be requeued.
```

check the system console or error log for messages of the following type:

```
12:52:37 !!! Print_Spooler Print_Device_TERMINAL_252 Open_Port: !Tools.
         Networking.Transport_Name.Undefined, from PC = #E400C, #C9
12:52:37 !!! Print_Spooler Print_Device_TERMINAL_252 A fatal error was
         detected on spooler device TERMINAL_252. The device is being
         disabled.
```

If present, the problem is it cannot find the network name which this device corresponds to. In this case, the name is `mktg_laser`. My problem was that I had deleted the `Tcp_Ip_Name_Server` file from which this name is found, and I did not have an entry in my `Transport_Name_Map` file.

```
   Device              Protocol           Characteristics  State   Classes
============ ============================ =============== ======= ========
TERMINAL_252 TELNET (mktg_laser ( 0, 23))   Laser_Comm    Enabled MLASER
```

The solution is to define the name somewhere, either the `Transport_Name_Map` or via the `Tcp_Ip_Name_Server`.

♦

INFO: Printing on another R1000	
Applicable to:	*Fix:*
References:	*Keywords:* Network, Printing, Queue, Remote
Originated from: Deg, Mam	*Revised by:*

This addresses the case when you have two machines, **A** and **B**. You have a printer connected to **B**, and you want to print on this device from machine **A**.

1. Add the following code to the `Initialize_Servers` procedure in `!Machine`. It is recommended that it be running on all machines in your network in case you want to move the printer to another machine.

```
begin
   Program.Run_Job
      (Program.Current ("!Tools.Rpc_Servers", "Queue_Service.Start",
                        Activity => "!Machine.Release.Current.Activity"),
       Context => "!Machine.Error_Logs",
       Options => "Output = !Machine.Error_Logs.Queue_Server_Output, " &
                  "Error = !Machine.Error_Logs.Queue_Server_Error, " &
                  "User = Network_Public, Password = (), " &
                  "Name =>(Print Queue Server)");
exception
   when others =>
      Error_Reporting.Report_Error
         (Caller => "!Machine.Initialize_Servers.Queue_Service.Start",
          Reason => Error_Reporting.Create_Condition_Name
                       ("Unhandled_Exception", Error_Reporting.Problem),
          Explanation => Debug_Tools.Get_Exception_Name (True, True));
end Print_Queue_Server;
```

2. Set up the classes on the machines.

   - If the default class on the remote machine **B**, i.e. `LP`, is to be used, and you want **A** to print to machine **B** by default then you must add the following to the `Initialize_Servers` procedure on machine **A**:

     ```
     Queue.Default("!!B");
     ```

   - If you don't want to use the remote machine's default class, then you must explicitly name the class. If the non-default class on **B** is `Laser`, then use:

     ```
     Queue.Default("!!B.Laser");
     ```

3. You will also need to make sure that `Network_Public` has `RC` access to `!Machine.Temporary`; also, the default ACL for `!Machine.Temporary` should be set to `Network_Public => RW`.

♦

TIP: Bogus queue devices	
Applicable to:	*Fix:*
References:	*Keywords:* Destruction, Printing, Queue
Originated from: Clp, Gbd	*Revised by:*

Problem: It is possible to create a useless, unremovable queue device, named `???`, by doing the following:

1. In the world `!Machine.Devices`, create a text file having whatever name you like. Suppose it's called `Abc`.

2. Then run `Queue.Add (Device => "Abc")`. This creates a bogus queue device. Data will not flow to such a *device* but at least you can still get rid of it by using `Queue.Remove (Device => "Abc")`.

3. Finally, delete the text file. Now you've *really* done it! `Queue.Devices` will show something like this:

```
Device              Protocol       Characteristics  State     Classes
============ ===================== =============== ======== ===========
???          XON_XOFF                               Disabled (none)
TERMINAL_250 TELNET (ps ( 0, 23))    Laser_Comm     Enabled  PLASER
```

The second device is genuine, the first is bogus. You can create many such devices and they will all be named `???`. A bogus device does no harm, but it's annoying because you can't get rid of it. `Queue.Remove (Device => "???")` will not get rid of it because `???` is not a valid device name. Neither will rebooting.

The only solution is to delete the files in `!Machine.Queues.Print_Spooler` then recreate your devices and classes from scratch. When the files are deleted all currently queued requests will be lost; actually there will still be files for them in the `!Machine.Queues.Print_Spooler.Temp` world but the files will just be garbage. The files in `Temp` should be deleted too.

◆

BUG: Pragma Page and the print spooler	
Applicable to:	*Fix:*
References:	*Keywords:* Pragma, Printing
Originated from: Pbk	*Revised by:*

Currently, the `pragma Page` in Ada units is ignored by the Print Spooler (`Queue.Print`), contrary to Rational's Appendix F for the R1000 (page 2) and in some sense to the RM B-2(9), although it might be argued that "our compiler is *not* producing a listing".

This is a bug. `Queue.Print` *should* care about `pragma Page` when printing an Ada unit.

The following "skin" around `Queue.Print` provides the missing functionality, by making use of the `Form` parameter `Pragma_Page_Mapping` while copying the Ada unit into a temporary text file. This form option is documented in the appendix F, p.41.

Note: This works also on a laser printer (using `Options => "Postscript..."`), with however the effect that, if option `Fancy` is used, the pragma will be listed on top of the new page; this does not quite comply with the RM specified semantic of the pragma which states that the text *following* the pragma should be on the next page [RM B-2(9)].

```
with Device_Independent_Io, Queue, Library, Time_Utilities;
procedure Print (Name : String := "<IMAGE>";
                 Options : String := "<DEFAULT>";
                 Banner : String := "<DEFAULT>";
                 Header : String := "<DEFAULT>";
                 Footer : String := "<DEFAULT>") is

   package Dio renames Device_Independent_Io;
   package Tu renames Time_Utilities;

   Temp_File : constant String :=
      "!Machine.Temporary.Spool_" &
      Tu.Image (Tu.Get_Time, Date_Style => Tu.Ada, Time_Style => Tu.Ada);

   S, D : Dio.File_Type;
   C : Character;

begin
   Dio.Open (File => S,
             Mode => Dio.In_File,
             Name => Name,
             Form => "Page_Pragma_Mapping = True");  --<<

   Dio.Create (File => D, Mode => Dio.Out_File, Name => Temp_File);

   while not Dio.End_Of_File (S) loop
       Dio.Read (S, C);
       Dio.Write (D, C);
   end loop;
```

```
   Dio.Close (S);
   Dio.Close (D);
   Queue.Print (Name => Temp_File,
                Options => Options,
                Banner => Banner,
                Header => Header,
                Footer => Footer);
   Library.Delete (Temp_File);
end Print;
```

♦

TIP: Telnet.Connect raises !Io.Io_Exceptions.End_Error

Applicable to:	*Fix:*
References:	*Keywords:* End Error, Exception, Network, Telnet
Originated from: Pam	*Revised by:*

If there is one (or more) blank line at the end of the `Transport_Name_Map`, `Telnet.Connect` fails with the following messages :

```
89/07/28 11:25:51 --- Telnet.Connect "etoile".
89/07/28 11:25:53 %%% An unexpected exception was raised in Telnet.
89/07/28 11:25:53 ... Connect_Initiate.
```

And, in the message window:

```
TELNET connection PAM.S_1.ETOILE.1 terminated due to unhandled exception
!Io.Io_Exceptions.End_Error
```

♦

BUG: Wollongong FTP server deficiency		
Applicable to: WINS/TCP version 4.0		*Fix:*
References:	*Keywords:* Ftp, Network, Server, Wollongong	
Originated from: Jmk, Rmf	*Revised by:*	

There is a deficiency in Wollongong's implementation of the FTP server; that's the FTPD program in WINS/TCP version 4.0, running on PC-DOS, with the Western Digital 8003 Ethernet card.

The symptom is that you can use the FTP utility program on the PC to put and get files from the Rational machine, but you cannot use the Rational `Ftp` command package to `Ftp.Put` or `Ftp.Get` anything from the PC. If you specify `Sent_Port => False`, you get an error message saying that it cannot connect to host 0.0.0.0. If you specify `Send_Port => True`, you get a error message saying that the port is privileged or restricted. These error messages appear in your Rational log window, but they are coming from the FTP server running on the PC. Like all such messages, they begin with a 3-digit numeric error code.

A workaround for this problem is to run the program below once each time the machine is booted, and then to specify `Send_Port => True` whenever attempting an FTP transaction with the PC running the Wollongong server.

```
declare
    C : Transport.Connection_Id;
    S : Transport_Defs.Status_Code;
begin
    loop
        Transport.Open (C, S, "TCP/IP");
        declare
            Socket : constant Transport_Defs.Socket_Id
                := Transport.Local_Socket (C);
        begin
            Transport.Close (C);
            exit when Byte_Defs.">=" (Socket (Socket'first), 4);
        end;
    end loop;
end;
```

This problem springs from two related bugs in Wollongong's code:

1. The server cannot form a data connection unless it first gets a `port` command specifying what address and socket number to use.

2. The server rejects any `port` command specifying a socket number that is `<= 1024` (also known as `(4,0)`). The server code was designed with the assumption that such ports can only be opened by programs running as superuser/root/system, not users.

◆

BUG: Ethernet hangs - one possible cause		
Applicable to: D1 TCP/IP Module V4.Aa		*Fix:* D2
References:	*Keywords:* Ethernet, Hang-Up, Network	
Originated from: Rjg	*Revised by:*	

This problem is fixed in D2. The symptom is that network connections are no longer possible. The following messages appear in the error log in fairly rapid succession (there may be some intervening messages) at the time things stop working:

```
12:21:07 --- Ethernet Controller_Status EXOS CODE 030A
12:21:27 --- Ethernet Controller_Status EXOS STACK TRACE: <A BUNCH OF NUMBERS>
12:21:37 --- Ethernet Controller_Status EXOS PANIC MODE
```

The problem is a bug in the software loaded into the Ethernet controller from the R1000. The workaround is to reboot the Ethernet controller by typing `Tcp_Ip_Boot` in a command window.

Note: During reboot of the Ethernet controller it tells you what version of the software is being loaded:

```
12:54:52 --- Ethernet Controller_Status TCP/IP Module V4.Aa
```

The above line shows version `4.Aa`, which has the bug. The new version is `4.Hb` - if you have this version, it is a new problem.
This problem should occur very rarely, so it is probably reasonable to wait for the new software in D2. If it becomes an issue, it is possible to update this software before D2.
♦

HINT: TCP/IP - DECnet transfers	
Applicable to:	*Fix:*
References:	*Keywords:* Decnet, Ftp, Network, Tcp Ip
Originated from: Mv	*Revised by:*

This is how to move files between an R1000 and a non-TCP/IP VAX. Consider the following:

We have one R1000 named **Mouse**, one VAX with TCP/IP named **VAX7** and one VAX without TCP/IP named **VAXU1**. Both VAXes are connected to DECnet. Then the following command will actually transfer a file from Mouse to **VAXU1** via **VAX7** (assume the file is a text file named HELLO):

```
Ftp.Put (From_Local_File => "HELLO",
         To_Remote_File =>
            "VAXU1""MVE password"":HELLO.TXT", -- password is for VAXU1
         Remote_Machine => "VAX7",
         Username => "MVE",
         Password => "password",                -- password is for VAX7
         Remote_Type => "VMS",
         Transfer_Type => Ftp_Defs.Ascii,
         Transfer_Mode => Ftp_Defs.Stream,
         Transfer_Structure => Ftp_Defs.File);
```

You can of course make a little bit simpler by using the *proxy login* feature of VMS (you don't have to specify the remote password).

♦

INFO: The package Remote	
Applicable to:	*Fix:*
References:	*Keywords:* Archive, Network, Remote, Rpc
Originated from: Mv	*Revised by:*

1. The `Remote` package is written on top of the Archive Server. It uses the `Rpc_Access_Utilities` package (located in `!Tools.Networking`) to supply the remote Archive Server with information about username and password (in order for the server to assume some other identity than `Network_Public`). In order to enable `Remote` to change the remote identity you need to:

   - Create a text file (in your home world for instance) with access information for the remote machine(s) on the form:

     ```
     HOSTNAME USERNAME PASSWORD
     ```

     For example:

     ```
     STAN      MVE       MAGIC
     SHEMP     MVE       SECRET
     OTHERS    RATIONAL <PROMPT>
     ```

     Note: `OTHERS` has to be the last.

   - Change session switch `Profile.Remote_Passwords` to point to the recently created text file.
   - Log out and in again (the pointer is cached at login).
   - Optionally, you can create a similar text file for your remote sessions (switch `Profile.Remote_Sessions`).

   The good thing about this is that it also works for `Archive.Copy`.

2. About the `Options` parameter, as far as I have been able to see in the code of the Archive Server, the options parameter for `Remote.Run` is *not* used at all.

♦

INFO: How FTP handles Ascii.FF	
Applicable to: D_12_1_1	*Fix:*
References: RFC 959	*Keywords:* Ascii, Ftp, Network, Tcp Ip
Originated from: Jmk	*Revised by:*

A change was made to FTP in Delta 2 concerning handling of `Ascii.FF` in ASCII mode. This change was *not* documented in the Delta 2 release note.

In Delta 2, if you copy a text file that contains page breaks from a R1000 via FTP with `Transfer_Type => Ftp_Defs.Ascii`, each page break will be transformed to a line break in the copy. Our FTP implements this transformation by transmitting each page break as `Ascii.CR & Ascii.LF` in the FTP interchange representation. In a Rational Environment file, a page break is stored as `Ascii.FF`. `Ftp_Defs.Ascii` is the default FTP transfer type; it corresponds to *ASCII* type and *non-print* format in the FTP protocol specification RFC 959.

To preserve page breaks, use `Transfer_Type => Ftp_Defs.Ascii_Telnet`. In the FTP standard this corresponds to `ASCII` type (RFC 959 section 3.1.1.1) with *Telnet* format control (section 3.1.1.5.2). With this Transfer_Type, each page break will be transmitted as `Ascii.CR & Ascii.LF & Ascii.FF`.

In Delta 1, a page break was transmitted via FTP as simply `Ascii.FF`, and the `Ascii_Telnet` mode was not supported.

The intent of this change is to support copying a file to a machine that does not recognize `Ascii.FF` as end-of-page, without losing information about end-of-line. In Delta 1, problems were experienced copying files to some Unix servers, in which the lines before and after a page break were combined into a single line, with significant loss of information. For example, an Ada fragment like:

```
-- The reactor core is too hot.
<end-of-page>
Shutdown_Reactor (Immediately);
```

was transformed into:

```
-- The reactor core is too hot. Shutdown_Reactor (Immediately);
```

If all machines in a customer's network support `Ascii_Telnet`, it may make sense to change the default type, by setting the session switch `Session_Ftp.Transfer_Type` to `Ascii_Telnet`.
♦

TIP: Line truncation when transfering files with FTP		
Applicable to:	*Fix:*	
References: CSR 1597, 2999, 3129		*Keywords:* Ftp, Ibm, Truncation
Originated from: Vnv	*Revised by:*	

When `Ftp.Put` or `Ftp.Store` is used to transfer files from an R1000 to an IBM (and perhaps other hosts...) the lines in the file get truncated at column 80.

- Past workaround

  Limit the line length to 80 characters on the Rational side:

  1. Use the library switch:

     ```
     Format . Line_Length          : Line_Range      := 80
     ```

     which would catch everything but long variable/path names.

  2. Then postprocess these files on the R1000 with a program that detects and flags what the previous step missed (not supplied here).

  3. Correct the flagged items by hand.

  4. Transfer the files.

- New workaround

  This one has recently been found and will work for systems whose networking software accepts a command that specifies what the maximum line length of the files to be transfered is. The one case where this has been used is an IBM 3090, running Sparticus 3.0 networking software.

  A program (not supplied here) that reads the files to be sent and determines the maximum line length is required to completely automate this process.

  Below is the code used to specify the maximum line length and send the files:

```
with File_Transfer;
with Ftp;
with Ftp_Defs;
with Ftp_Profile;
with Log;
with Profile;

procedure Send_File (Local_File : String := "<IMAGE>";
                     Remote_File : String := "";
                     Max_Line_Length : Positive) is

   procedure Get_Response
               (Connection : File_Transfer.Connect_Id;
                Command    : String) is
```

```
        -- This is a highly functional implementation:
        -- a simpler workable implementation is possible.
        Answer : Ftp_Defs.Status_Code;
        Count : Natural;
        Max_Response_Line : Natural := 512;
        Line : String (1 .. Max_Response_Line);

        function "=" (X, Y : Ftp_Defs.Status_Code) return Boolean
            renames Ftp_Defs."=";

        function To_Msg_Kind (X : Natural) return Profile.Msg_Kind is
        begin
            case X / 100 is
                when 4 | 5 =>
                    return Profile.Error_Msg;
                when others =>
                    return Profile.Note_Msg;
            end case;
        end To_Msg_Kind;

    begin
        while not File_Transfer.End_Of_Response (Connection) loop
            File_Transfer.Read_Response (Connection, Line, Count);
            Log.Put_Line (Line (1 .. Count),
                          To_Msg_Kind
                             (File_Transfer.Most_Recent_Response_Code
                                 (Connection)));
        end loop;

        File_Transfer.Command_Status (Connection, Answer);
        if (Answer = Ftp_Defs.Successful) then

            Log.Put_Line (Command & " completed successfully.",
                          Profile.Positive_Msg);
        else
            Log.Put_Line (Command & " failed with status = " &
                          Ftp_Defs.Status_Code'Image (Answer),
                          Profile.Negative_Msg);
        end if;
    end Get_Response;


begin
    Ftp.Connect (To_Machine => Ftp_Profile.Remote_Machine,
                 Auto_Login => Ftp_Profile.Auto_Login,
                 Username => Ftp_Profile.Username,
                 Password => Ftp_Profile.Password,
                 Account => Ftp_Profile.Account,
                 Remote_Directory => "",
                 Remote_Roof => Ftp_Profile.Remote_Roof,
                 Remote_Type => Ftp_Profile.Remote_Type,
                 Transfer_Type => Ftp_Profile.Transfer_Type,
                 Transfer_Mode => Ftp_Profile.Transfer_Mode,
                 Transfer_Structure => Ftp_Profile.Transfer_Structure,
                 Send_Port => Ftp_Profile.Send_Port_Enabled,
                 Response => Profile.Get);
```

```
      File_Transfer.Send_Site_Command
         (Connection => Ftp.Current_Connection,
          Argument => "lrecl(" & Positive'Image (Max_Line_Length) & ")");

      Get_Response (Ftp.Current_Connection, "SITE");

      Ftp.Store (From_Local_File => Local_File,
                 To_Remote_File => Remote_File,
                 Append_To_File => False,
                 Response => Profile.Get,
                 Remote_Type => Ftp.Current_Remote_Type,
                 Account => Ftp_Profile.Account);

      Ftp.Disconnect (Response => Profile.Get);
   end Send_File;
```

◆

TIP: When a system-defined group gets deleted		
Applicable to:	*Fix:*	
References:	*Keywords:* Acl, Destruction, Group, System	
Originated from: Jgp, Rjg	*Revised by:*	

A simple way this can happen is if a user with the same name as a group (such as `Privileged`) gets created and deleted.

- If one of the following group is deleted:

```
0  PUBLIC
1  NETWORK_PUBLIC
2  PRIVILEGED
3  RATIONAL
4  MAILER
5  SPOOLER
6  SYSTEM
```

  Rebooting the system will recreate the group, but no users will be in that group - so users must be added to the group.

  Note: The recreated group *will not* appear in the name map (from `Show_Groups`) until the following reboot. This is because the name map is evaluated before the group is recreated during the first reboot. The behaviour should otherwise be normal after the first reboot. In one case however, not only did the names of the groups not appear in the ACLs after the first reboot, but the ACLs did not work until the second reboot. In this case, `Public` and `Network_Public` were deleted, and no users could perform any commands until after the second reboot, and after all users were added back into the recreated groups.

- If another *special* group (such as `Operator`) is deleted, it can be recreated, but will no longer have all its special powers. You can still log in as `Operator`, but if `Access_List.Display` is run on objects that used to have `OPERATOR => ...`, it now returns:

```
<Unknown 18> => ...
```

  This is because `Operator` does not have a fixed group number like `Public`, `Network_Public` etc, and rebooting does nothing more that create the user `Operator` if not present. Rebooting will not restore privileges or operator capability to `Operator`, and it will not fix the `<Unknown #>` ACLs. You thus have to:

  1. Give `Operator` access to `!Machine.Operator_Capability`
  2. Add `Operator` to group `Privileged`.
  3. Write a routine that visits all objects on the system checking the ACLs on each, and, if `<Unknown 18>` has access to an object, then give `Operator` the same ACLS using `Acl.Set`.

◆

TIP: Code_Segment exception Storage_Error in manager task

Applicable to: D1	*Fix:* improved in D2
References:	*Keywords:* Code Segment, Exception, Hang-Up, Loaded Procedure
Originated from: Vnv	*Revised by:* Rjg

At the operator console, one sees:

```
*** CODE_SEGMENT Exception Storage_Error, from PC = #nnnnn,#nn (in
    manager task)
*** Calling task (16#nnnnnnnn#) will be stopped in wait service
```

At this point two things can happen:

1. The system may hang. It will do so if the suspended task runs under VPID 4 (the last two hex digits of the task are `16#04#`).

2. If the system doesn't hang, users will go into an infinite running state when they promote Ada units.

The problem is that the machine has run out of code segments. Code segments are consumed by loaded procedures, especially those that have a lot of units in their closure. Code segments are limited by disk volume, i.e. on each volume, X amount of code segments are available. Code segments are only reclaimed at reboot. There is no way to find out how many code segments are used for objects already created.

Rebooting the machine will recover some code segments and restore the machine to normal operating state.

Work-around:

1. Delete all old, unused, duplicate loaded procedures.

2. Spread loaded procedures out evenly over all volumes. To get a very rough idea of their current distribution run:

```
Lib.Space (For_Object => "!??'c(load_proc)",
           Levels => 2,
           Recursive => True,
           Each_Object => True,
           Each_Version => True,
           Space_Types => True,
           Response => "<PROFILE>",
           Options => "");
```

this will list all loaded procedures, in a listing that will look like:

```
NAME                          VERSION  VOL  TOTAL   ADA  ATTR  IMAGE
===========================   =======  ===  =====  ====  ====  =====
!COMMANDS.ARP_SHOW            *V(1)      1     19    10     7      2
!COMMANDS.ARP_UNDEFINE        *V(1)      1     19    10     7      2
!COMMANDS.DOCUMENT_BUILDER    *V(2)      1     24    12     9      3
```

```
!COMMANDS.ENCAPSULATE        *V(1)    1     19     10    7      2

=== Volume Totals:                    1   1501    777  559    165
                                      2    210    105   82     23
                                      3    168     90   60     18
                                      4    137     71   51     15

=== Grand Total:                    all   2016   1043  752    221
```

If most are on one volume, as is the case for volume 1 in the example above, move off some to other volumes.

Note: The sizes listed in the total column do *not* have any direct correspondence to number of code segments used. This is only a ball park guess. Since there is no direct correspondence between sizes above and code segments consumed there still is a chance that if most code segments are used up in a few large procedures then the redistribution might fail to solve the problem.

3. You *must* reboot the machine to reclaim the code segments.

Note: In D2, code segments are shared wherever possible. But since code segments are still a finite resource one can conceivably run out of them even in D2. Also not that there has been no improvement to the error message, so if it does happen it will just as difficult to diagnose.

♦

TOOL: A tool to remove a group from an ACL	
Applicable to:	*Fix:*
References:	*Keywords:* Acl
Originated from: Phil	*Revised by:*

Here is a tool for deleting a group from an ACL or a bunch of ACLs:

```
with String_Utilities;
with Access_List;
with Access_List_Tools;
with Directory;
procedure Remove_Group_From_Acl (G : String; Objects : String) is
Iter : Directory.Naming.Iterator;
Name_Status : Directory.Naming.Name_Status;
procedure Set_Acl (To_List : Access_List.Acl := "Network_Public => RWCOD";
                   For_Object : Access_List.Name := "<SELECTION>";
                   Response : String := "<PROFILE>")
    renames Access_List.Set;
begin
Directory.Naming.Resolve (Iter, Objects, Name_Status);
while not Directory.Naming.Done (Iter) loop
    declare
        Name : constant String := Directory.Naming.Source_Name (Iter);
        Old_Acl : constant String :=
           "," & Access_List_Tools.Get (Name) & ",";
        Pos1 : Natural := String_Utilities.Locate
                              ("," & G & "=>", Old_Acl, Ignore_Case => True);
        Pos2 : Natural;
    begin
        if Pos1 /= 0 then
            Pos2 := String_Utilities.Locate
                       (Fragment => ",",
                        Within => Old_Acl (Pos1 + 1 .. Old_Acl'Last));
            if Pos1 = Old_Acl'First then
                Set_Acl (Old_Acl (Pos2 + 1 .. Old_Acl'Last - 1), Name);
            elsif Pos2 = Old_Acl'Last then
                Set_Acl (Old_Acl (Old_Acl'First + 1 .. Pos1 - 1), Name);
            else
                Set_Acl (Old_Acl (Old_Acl'First + 1 .. Pos1 - 1) &
                         Old_Acl (Pos2 .. Old_Acl'Last - 1), Name);
            end if;
        end if;
    end;
    Directory.Naming.Next (Iter);
end loop;
end Remove_Group_From_Acl;
```

♦

PROBLEM: Maximum number of entries in an ACL	
Applicable to:	*Fix:* Deferred
References:	*Keywords:* Acl, Maximum
Originated from: Jgp	*Revised by:*

An ACL can hold 7 entries. The problem is how to get the seventh entry into the ACL. There is a bug in `Acl.Set` and `Acl.Add` which prevent 7 entries in an ACL; the maximum they will allow is 6 entries. However, `Access_List_Tools.Set` does not use the same check and will allow setting an ACL that has 7 entries. The following procedure could be used to replace `Acl.Set`.

```
with Access_List_Tools;
with Io;
with Simple_Status;
procedure Acl_Set (To_List : String;
                   For_Object : String := "<SELECTION>";
                   Response : String := "<PROFILE>") is

    Status : Simple_Status.Condition;
begin
    Access_List_Tools.Set
       (To_List => To_List, For_Object => For_Object, Status => Status);
    if Simple_Status.Error (Status) then
        Io.Put_Line (Simple_Status.Display_Message (Status));
    end if;
end Acl_Set;
```

♦

PROBLEM: ACLs referencing a deleted group	
Applicable to: D1, D2	*Fix:*
References: CSR 3350	*Keywords:* Acl, Group
Originated from: Vnv	*Revised by:* Ams

- Deleted groups can prevent you from adding an entry to an ACL. When using `Acl.Add`, you get the message:

```
--- Access_List.Add (To_List => "Network_Public => RWCOD",
... For_Object => "<SELECTION>");
*** <File>: Bad group name : deleted group:
... <UNKNOWN 41>.
+++ Changed access list for  0 objects.
+++ Failed to change access list for  1 objects.
```

  This is because the ACL contains references to a deleted group. When you use `Acl.Display`, you get:

```
<File>            : <UNKNOWN 41>=>RCOD,NETWORK_PUBLIC=>RCO
```

  The solution is to use `Acl.Set` to replace the whole ACL.

- In D2, a similar problem may arise with `Cmvc_Access_Control`. If a group mentioned in a CMVC ACL gets deleted, and the Access List Compaction daemon is run, you still have the deleted group (i.e. the `<UNKNOWN 41>`) in the CMVC ACL but not in the regular ACL.

  Running:

```
Cmvc_Access_Control.Check (Repair_Inconsistencies => True);
```

  will remove the bogus group. You can run this before or after the Access List Compaction daemon to remove the deleted group reference from the CMVC ACL.

◆

TIP: Unhandled exceptions editing Ada units	
Applicable to:	*Fix:*
References:	*Keywords:* Acl, Ada Unit, Creation, Exception, Switch
Originated from: Cbh, Lhrp	*Revised by:*

If you get `Unhandled exception` or `Other error` when attempting to create an Ada unit or to open an insertion point in an existing Ada unit, and if the library switches cannot be displayed, it may be that `!Machine.Switch_Definitions` has the wrong ACL.

On a customer's machine this ACL was `Network_Public => R, Operator => RW`, whereas it should be `Network_Public => RW`. The behaviour that follows in this case is:

1. Users can not display library switch files. Only the titles of the columns are displayed without any separating spaces.

2. After users tried to display library switches, they can no longer do operations like creating an Ada unit. The result is an unhandled exception. From this point on they are not even able to create Ada units in any library (even without associated switches).

3. The operator can display the library switches. After he has done this, normal users find all their problems solved after they logout and login.

4. The whole thing starts again after each boot of the machine.

Theory:

`!Machine.Switch_Definitions` is used when consulting library switches. After the first user did this, the machine caches this information until the next boot. Therefore only the first user consulting library switches needs RW access to `!Machine.Switch_Definitions`. You probably need to logout and login to be able to access the cached information.

Once you failed to consult library switches, you get into this mode of getting funny errors where ever you want to do something involving library switches until you logout.

Note: You can create the same behavior by deleting all access to a library switch file itself.
♦

INFO: ACLs needed to upgrade to D2	
Applicable to: D2	*Fix:*
References:	*Keywords:* Acl, Upgrade
Originated from: Smp	*Revised by:*

Before upgrading to D2, ensure that `!Machine.CG_Data` and its contents have not been frozen and have the correct ACL values:

```
System => RW
```

You may use `Set_Universe_Acls` for setting this in a restrictive manner if site conditions require it.

In addition, verify that `!Compiler_Interface` is not frozen and that `!Compiler_Interface.Subsystem_Interface` has the following ACL:

```
Public => RW
```

♦

INFO: How to change a password		
Applicable to: D1	*Fix:*	
References:	*Keywords:* Password	
Originated from: Deg	*Revised by:*	

There is a new undocumented feature in D1 that will allow you to change a password by suppling an authorization code for the old password. This is extremely usefull when the operator password is forgotten.

The three methods to change a password are:

```
Op.Change_Password ("Username", "Old_Password", New_Password");
Op.Change_Password ("Username", "Operators_Password", New_Password");
Op.Change_Password ("Username", "Authorization_Code", New_Password");
```

Call the Response Center to get the authorization code. The Cluster Id and the current date on the machine is needed.

◆

TIP: Setting password policies on existing machines	
Applicable to:	*Fix:*
References:	*Keywords:* Password
Originated from: Rjg	*Revised by:*

There is a very simple solution to the problem of inadvertently locking out users when first implementing a password policy.

- First of all, while first setting things up, do the `Op.Set_Password_Policy` manually each time the machine is booted; do not add it to `!Machine.Initialize_Site` until you're sure it's working the way you want. This way you can always back out of a horrible mistake by rebooting the machine.

- For a limited time, however long you want to allow users to update their passwords (perhaps a couple of weeks), do the following: for example, assume that eventually you want to set things up such that `Min_Length = 7, Change_Warning = 86, Change_Deadline = 100.`

```
declare
    Min_Length : constant Natural := 7;
    Warning_Age : Op.days := 86;      -- Set this up the way you eventually
                                      -- want.
    Deadline_Age : Op.Days := 1000;   -- Longer than the machine has been
                                      -- around, so that no one will be
                                      -- affected. NOTE: Do not use
                                      -- Operator.Days'last - there is a
                                      -- bug with this!
begin
    Op.Set_Password_Policy (Minimum_Length => Min_Length,
                            Change_Warning => Warning_Age,
                            Change_Deadline => Deadline_Age);
end;
```

  As long as you leave it this way, users with passwords older than 86 days will get the following message when logging on:
  `Your password is in danger of expiring; it should be changed.`
  but no passwords will expire!

- When you have waited long enough, set up the password policy with the desired change deadline. Users with expired passwords will now be locked out, but they have been warned!

- When everything has settled down, add the call to `Op.Set_Password_Policy` in `!Machine.Initialize_Site` so that you don't have to do it by hand after every reboot.

♦

PROBLEM: DDB daily daemon gets Restore_Failed exceptions	
Applicable to:	*Fix:*
References:	*Keywords:* Daemon, Exception
Originated from: Hjl, Rjg	*Revised by:*

The errors look like this:

```
00:18:20 +++ Ddb Started
00:20:44 +++ Dependency_Data_Base RESTORE_FAILED Exception during
         perform_restore
00:20:52 +++ Dependency_Data_Base RESTORE_FAILED Exception during
         perform_restore

... possibly hundreds of identical messages...
00:50:56 +++ Ddb Completed old = 1234; new = 1177 (pages)
```

The Ddb message is a warning message. we don't think it's a serious problem. If it happened once I would ignore it. There is nothing more that could be done with it at this time. If it happens all the time we would like to understand better why it is happening, although, looking at the code in the Ddb there doesn't seem to be a lot that could be done to understand it short of having a debugger running while it is happening.

◆

PROBLEM: DDB daily daemon gets Restore_Failed

Rev. 0

TIP: None of the daemons are running	
Applicable to:	*Fix:*
References:	*Keywords:* Backup, Daemon, Kill
Originated from: Jgp	*Revised by:*

If you have a system where none of the daemons are running, check to see if backup kill mode is disabled and a backup job is running. To check, run the function `Backup_Killing_Enabled` in `!Tools.Disk_Daemon`, which returns a boolean value.

We had a case where the operator running the backup disconnected during a tape mount request. Since backup kill mode was disabled, the idle backup job prevented any of the daemons from running for two days until it was discovered and killed. (If backup kill mode had been enabled, the garbage collector would have killed the backup job.) Normally, backup kill mode is enabled when the system boots.

To enable or disable backup kill mode, use the procedure `Set_Backup_Killing`, also in `!Tools.Disk_Daemon`.

♦

TIP: Error log warnings to pay attention to	
Applicable to:	*Fix:*
References:	*Keywords:* Daemon, Error Log
Originated from: Deg	*Revised by:*

In general, any problems/warnings that prevent any manager from compacting successfully should be brought to Rational's attention. All of the managers check their data for consistency as they do compaction (the main reason we run the daily daemons). They will refuse to compact their data (make a smaller copy of it) if they find any problems. This is just a delaying tactic, since there is probably little need to bring the machine to its knees in this case. If other operations touch this inconsistent data, the machine will stop.

◆

TIP: Disk daemon backs off after Stop_Jobs has been reached	
Applicable to:	*Fix:*
References:	*Keywords:* Backoff, Daemon, Disk
Originated from: Gpa, Hts, Jim	*Revised by:*

Backing off is the process whereby garbage collection tries not to be obtrusive in its operation. There are cases where the disk daemon will continue to back off even though `Stop_Jobs` has been reached. This is indicated by three symptoms:

- The amount of time backed off shown by `Show_Gc_State` continues to increase.
- There are only small bursts of CPU and disk activity every few minutes or so indicated by the control panel LEDS.
- The garbage collection takes *hours* to complete even though not much is happening on the machine.

There are two reasons why this situation may arise:

1. If a load remains on the machine, despite jobs being stopped, this might get in the way. Such a load may be due to higher priority activities, such as compaction daemons, backup, etc.
2. A non-standard backoff algorithm could cause garbage collection to backoff indefinitely in the presence of jobs withheld by the MTS (note that `Job.Disable` causes jobs to be considered *withheld* by the MTS).

   Theory: During the `Traversing_Virtual_Memory` phase, the collector follows roughly the following algorithm, taken from `!Tools.Disk_Daemon'Spec`:

```
loop
    Do_About_500_Milliseconds_Of_Work;
    if Run_Load > Stop_Run_Load or
       Withheld_Load > Stop_Withheld_Load then
       loop
          delay 30.0;
          exit when Run_Load < Start_Run_Load and
                    Withheld_Load < Start_Withheld_Load;
        end loop;
    end if;
end loop;
```

   The loads for the stop tests are sampled over the last minute. The load for the restart tests are sampled over the last 5 minutes.

   The parameters `Start_@_Load` and `Stop_@_Load` may be changed/queried by subprograms declared in `Disk_Daemon`. The values supplied for these parameters should be MTS loads, multiplied by 100. For example, 125 means an MTS load of 1.25.

   The `Disk_Daemon.Use_Standard_Backoff_Algorithm` procedure will revert the collector to using its built-in backoff algorithm.

Explanation: Reaching `Stop_Jobs` will increase the withheld load by at least 1 for each running job. The problem is that the backoff algorithm wants to use withheld load as an indicator that the MTS has stopped running a job (because the machine is being oversubscribed). It is unfortunate that explicitly disabling jobs (for instance via `Job.Disable`), causes the job's tasks to be considered withheld.

Conclusion: Customers are encouraged to just live with the standard garbage collection parameters and behavior: the interaction between parameters is often surprising (even to implementors and users that have been bitten before).

If however a customer wants to supply a non-standard backoff algorithm, the withheld loads should be set to `Integer'Last`.

♦

INFO: What to do when you run out of action ids	
Applicable to:	*Fix:*
References:	*Keywords:* Action, Daemon, Job
Originated from: Vnv	*Revised by:*

If you can't free up any by doing the actions below, the rebooting is the only other alternative. There is a maximum of 1024 actions ids. Some are only freed at reboot, so if after the `Action` daemon has run there are still more than 500 or 600 in progress, one should reboot the system. This is one reason weekly reboots are recommended.

1. To find out which job has all the ids, do:

```
Show_Locks;
```

This gives you the following output:

```
Actions in progress =  266
Actions abandoned   =  0
Next Action Id      =  0

Job  #Actions      Job name
---- -------- --------------------
  4       82     System
  5       25     Daemons
173        9     [KAM.S_1 Editor]
```

The same information may be obtained from the Kernel by typing:

```
Kernel: Job_Names
```

This produces output like:

```
Threshold [1]:
Job CPU%      Root    Job Seg Acts   Name
--- ---- -------- --------- ----  --------------------------
  4    6        0  6C01C503  100 System
  5    0        0  D6842901   25 Daemons
151    0    5BC97  5E99AD02    0 SOFTWARE.ENVIRONMENT % WHAT.TIME
155    0    2409B  5E976502    4 "!Local".Top
159    0        0  6C0F6903    2 [TOP.MAM Editor]
161    0        0  6C0FA903    0 [TOP.MAM Command]
```

2. Disable or kill the job holding the action ids (using `Job.Disable` or `Job.Kill`).

3. Run the `Action` daemon:

```
Daemon_Run("Action");
```

If there isn't enough action ids to do the above, try to get some by having people logout using the break key 3 times; this sometimes frees up enough.

♦

TIP: Running out of group IDs

Applicable to:	*Fix:*
References:	*Keywords:* Action, Exception, Group
Originated from: Jim, Rjg	*Revised by:*

1. Symptoms

Environment logins hang completely; even editor operations are dead. The console is alive except console command interpreter. This is preceded by the following console error message:

```
09:21:24 *** GROUP Exception <Exception: Unit = 2297540, Ord = 1>, from
             PC = #32EC0A, #110E (in manager task)
         *** Calling task (16#14A35804#) will be stopped in wait service
```

`Op.Create_User` stimulates the problem.

2. Explanation

All this is caused by an exception being raised in the group object manager. This would certainly be involved in any call to `Op.Create_User`. It will most certainly stop logins; no references to objects of class `group` is possible while the manager is stopped. The problem is that the environment is running out of group id's.

3. Fix

In order to recover, you need to do the following:

```
Daemon.Set_Access_List_Compaction;
Daemon.Run ("ADA");
Daemon.Run ("FILE");
Daemon.Run ("DIRECTORY");
Daemon.Run ("GROUP");
Daemon.Set_Access_List_Compaction (On => False);
```

4. How to avoid the problem

It is reasonable to run the above steps every few months.

◆

TIP: Servers that do not start from !Machine.Initialize	
Applicable to:	*Fix:*
References:	*Keywords:* Exception, Server, Status Error
Originated from: Vnv	*Revised by:*

If may happen that a server starts fine if you initiate it from a command window, but will not start if you put it into `!Machine.Initialize`. This may for instance be the case with a customized PDL registration.

The problem is that many servers are trying to write a success message to the message window. But there is no message window available when the system boots so it would display:

```
!!! unhandled exception !Io.Io_Exceptions.Status_Error (not open)
```

in the error logs.

The solution is to specify the messages to be sent to `!Machine.Devices.Nil` by writing the following:

```
Program.Create_Job
    (Registration_Command,
     Context => Get_Context (Registration_Command),
     Options => "Input | Output | Error => !Machine.Devices.Nil" --<<
     Job => Job_Id,
     Status => Status);
```

♦

HINT: Ideas on solving memory-based performance problems	
Applicable to:	*Fix:*
References:	*Keywords:* Memory, Performance, Scheduler, Working Set
Originated from: Cbh	*Revised by:*

1. Try changing the foreground time limit to something smaller, which will push longer jobs into the background faster. Try 3 to 5 minutes.

2. At the same time, increase the maximum foreground budget to 1000. This keeps new jobs from being withheld for a full second. Should favor small commands, such as editors.

3. Try the same scenario without the resource-consuming jobs. Jobs using many pipes may be a real memory hog.

4. Determine what the memory requirement of the resource-consuming jobs is. On an otherwise idle machine, run these jobs, let them get cooking, then run `Scheduler.State` and `What.Users`. This allows you to add up how much memory each component job is using. An alternative is to run `Save_Performance_Data` while the jobs are running. If you try these, please send data to CBH.

5. While you are looking at this, determine how many separate jobs make up the application, including some description of their dynamic behavior.

6. Try turning memory scheduling off while running the scenario. This might help or hinder; we don't know.

7. Verify the scheduling parameters are reasonable. Please send a copy to CBH.

8. Try setting the `Max_Background_WSL` down to 2000 or so. This might have a big impact if the application is many jobs. Alternatively, have the jobs set their own working space limit to 2000 or so. This suggestion assumes the application is multiple jobs, and not some big monolithic single job. The advantage of the second approach is that the other background jobs, such as compilation, will still have their full memory allotment.

9. Irrespective of the above, run `Save_Performance_Data` on a machine exibiting slow response, and send it to CBH. Let the collection run for around 5 minutes, sampling every 20 seconds. Please be sure the machine is really running poorly when you do this.

♦

INFO: Unauthorizing a product	
Applicable to: D1	*Fix:*
References:	*Keywords:* Authorization
Originated from: Mam	*Revised by:*

This cards describes how to unauthorize a product that is already authorized.

- Call the Response Center and get an authorization code for that product. The expiration date will be the date on which you wish the product to expire. The `Product_Registration` procedure expects a expiration date which has not yet passed.
- Run the `Product_Authorization.Registration` procedure with the product name, the code as given by the Response Center and the expiration date. The expiration date given must match the one used to create the authorization code.
- Reboot the machine after the product has expired. If the product is to expire today, then reboot tomorrow.

Note: The expiration of a given product had resulted in the expiration of other none related products. This has been fixed in D1.

♦

INFO: System.Assertion_Error while promoting a unit with a CDF	
Applicable to:	*Fix:*
References:	*Keywords:* Assertion Error, Cdf, Exception, Promote
Originated from: Jlk	*Revised by:*

When trying to promote units in a CDF view, you get the message:

```
*** Messages generated while promoting <UNIT>
*** Unhandled exception occurred during assembly: !Lrm.
... System.Assertion_Error<Space= 0, Index= 135> raised at
... #181903, #88.
*** Parent already has a child with that name.
```

The problem is that the Ada unit is in the `Source` state but there exist *pointy* files named `.<Exe>`, `.<Obj>`, etc. This may occur if you use `Archive.Copy` to move the contents of a CDF view: the Ada units are demoted to `Source`, but these pointy files remain. During normal promotes and demotes these files are deleted and re-created.

Should you encounter this error you can quickly solve it by deleting the pointy files:

```
Compilation.Destroy (Unit => "@.<@>",
                     Threshold => 9999,
                     Limit => "<WORLDS>",
                     Response => "<PROFILE>");
```

You can then promote the units.

♦

TIP: Demoting predefined units in CDFs	
Applicable to:	*Fix:*
References: CSR 3828	*Keywords:* Cdf, Demote
Originated from: Pbk	*Revised by:*

A customer is *not* supposed to demote the contents of `!Targets.<Some_CDF>.Lrm` that contains units like package `System`, `Calendar` and the dreadful generic units, `Unchecked_Conversion` and `Unchecked_Deallocation`.

But, well, you'll always find someone that will try to do it. In this case, simply repromoting those units to Coded does not do you good. The cross-compiler does not treat them like other units: they are registered and cached somewhere. The net result is that many references to the predefined units will just lamentably fail with horrible error messages like:

```
!!! prompt in statement list
*** Unexpected Diana: Diana_Typing.Get_Discriminants DN_CONSTRAINED
%%% in Decl_Gen.Gen_Decl - constraint_error <Space= 0, Index= 224> raised
    at #F8F016, #479 while coding procedure FREE is new
    UNCHECKED_DEALLOCATION (CONTROL_BLOCK, ...);
```

The way to proceed after these units have been demoted is the following:

- Start the cross-compiler (e.g. `M68k.Start`)
- Install the contents of `!Targets.<Some_CDF>.Lrm` (and only this world)
- Restart the compiler (yes, restart !)
- Code the contents of `!Targets.<Some_CDF>.Lrm`
- Only then proceed with any other compilations.

♦

BUG: CDF generates exception when coding unit with warnings	
Applicable to: MC68020_@ Rev5	*Fix:*
References:	*Keywords:* Binding, Cdf, Mc68K, Xlib
Originated from: Rfg	*Revised by:*

The following code generates expected warnings and an unexpected CDF exception.

```
function Regions_Bug_Variant return Integer is
   function Xcreateregion return Integer;
   pragma Interface (C, Xcreateregion);
   pragma Import_Function (Internal  => Xcreateregion,
                           External  => "_XCreateRegion",
                           Mechanism => Value);
begin
   return Xcreateregion;
end Regions_Bug_Variant;
```

This generates:

```
1: WARNING VALUE too many mechanisms provided
2: WARNING Function XCREATEREGION is interfaced but
does not have an importing pragma

REGIONS_BUG_VARIANT'Body changed to INSTALLED

Coding of REGIONS_BUG_VARIANT'Spec has begun on Stream 1
Coding of REGIONS_BUG_VARIANT'Body has begun on Stream 1

Promote failed - Code generation errors found
1: EXCEPTION_HANDLED in Stmt_Gen.Gen_Stmt -
!Lrm.System.Assertion_Error<Space=0, Index= 135>
raised at #E0901, #C7A while coding return XCREATEREGION;
```

Removing the `Mechanism => Value` parameter allows the unit to be coded.

♦

BUG: CDF generates exception when coding unit with

Rev. 0

BUG: Mispelled interface name in Runtime_Interface package	
Applicable to: MC68020_Bare Rev3 and Rev5	*Fix:*
References:	*Keywords:* Mc68K
Originated from: Rfg	*Revised by:*

`!Targets.Mc68020_Bare.Target_Interface.Runtime_Interface'Spec` incorrectly specifies the internal name of the `Transition_Task_State` routine.

```
pragma Import_Procedure (Internal  => Transition_Task_State,
                         External  => "__TRANSTION_TASK_STATE", --<< Missing 'I'
                         Mechanism => (Value, Value));
```

`__TRANSTION_TASK_STATE` **should be** `__TRANSITION_TASK_STATE`.

A main program trying to reference this procedure will have an undefined symbol at link time.

The workaround is as follows:

- Demote `Runtime_Interface'Spec` to installed.

  An `Op.Enable_Privileges` may be required because of ACLs.

- Incrementally edit the pragma `Import_Procedure` to have the correct external name.
- Promote the specification to coded.
- Promote the body to coded. (The body was automatically demoted when the specification was).

♦

BUG: Mispelled interface name in Runtime_Interface

Rev. 0

BUG: CDF creates overlapping objects		
Applicable to: MC68020 Bare Rev5		*Fix:*
References:	*Keywords:* Cdf, Mc68K	
Originated from: Seh	*Revised by:*	

Beware of optimization_level 0 !

The following package exhibits a code generation error. The library switch `Cross_Cg.Optimization_Level` is 0; defaults are used for the other switches. The problem is that two objects, `Share_1` and `Share_2`, are sharing memory (i.e. their representation is overlapping). When one gets modified the other one too.

```
package body Pack is
   Bug : exception;
   Filler : array (1 .. 10) of Boolean;

   Shared_1 : array (1 .. 10) of Short_Short_Integer := (others => 0);

   type Validity_Status is array (1 .. 3) of Boolean;
   All_Valid : constant Validity_Status := (others => True);

   type System_Time is
      record
         H : Long_Float;
         S : Boolean;
      end record;

   type Position is
      record
         X1, X2 : Long_Float;
      end record;

   type Data is
      record
         D1, D2 : System_Time;
         The : Position;
         Is_Valid : Validity_Status := (others => False);
      end record;

   Shared_2 : Data;

   procedure Action is
   begin
      Shared_1 (9) := 0;
      Shared_2.Is_Valid (1) := True;  -- >> the problem <<
      if Shared_1 (9) /= 0 then
         raise Bug;
      end if;
   end Action;

end Pack;
```

The assembly listing of `Pack'Body` shows also the problem; `Shared_1 (9)` and `Shared_2.Is_Valid (1)` both are referencing `STATIC+21`:

```
          .STATEMENT   131073        ; statement # 1
                                     ; SHARED_1 (9) := 0;

000000C2    423900000000             CLR.B    (STATIC+21)
          .STATEMENT   131074        ; statement # 2
                                     ; SHARED_2.IS_VALID (1) := TRUE;

000000C8    13FC000100000000         MOVE.B   #1,(STATIC+21)

...

                                     BUG.1           EQU STATIC_INIT
                                     FILLER.AS.2     EQU STATIC+3
                                     SHARED_1.AS.3   EQU STATIC+13
                                     ALL_VALID.AS.6  EQU STATIC
                                     SHARED_2.9      EQU STATIC+21
```

Note: The problem disappears with `Optimization_Level` set to 3 (the default).

♦

TIP: Design.Check_Consistency, Clean_Caches, and Fix_Caches	
Applicable to:	*Fix:*
References:	*Keywords:* Design Facility, Exception
Originated from: Vnv	*Revised by:*

There has been some confusion on when which of the above should be used... `!Commands.Fix_Cache` should be used when document generation for a CSCI fails with either:

```
Unexpected exception in Start
```

or the unhandled exception `Write_To_Read_Only_Page` is raised.

In all other cases, `Design.Check_Consistency` is the cleaning mechanism to start with. This catches most problems with the least cost.

If `Design.Check_Consistency` fails then try:
`!Tools.Design.Release.Utilities'Spec_View.Units.Clean_Caches`, this routine catches some of the cases `Check.Consistency` misses, but at a greater cost (demoting and repromoting all units in the view).

In the Bug Fix Release of the 2167 product, and the 2167A product, `Design.Check_Consistency` will do the right thing.
◆

HINT: RDF - controlling the phase on a unit basis	
Applicable to:	*Fix:*
References:	*Keywords:* Ada Unit, Design Facility, Phase
Originated from: Pbk, Rjs	*Revised by:*

The default phase handling is on a view basis, but a particular customization can override this default relatively easily. The key to this lies in the generic formal subprogram `!Tools.Design.Pdl.Annotation.Description.Traversal_Pre_Op`:

```
with procedure Traversal_Pre_Op
                  (Current_State       :        Unit_State;
                   Goal_State          :        Unit_State;
                   Mode                :        Analysis_Mode;
                   Phase               : in out Phases;
                   Root_Element        :        Ada_Program.Element;
                   Major_Elements_Only : out    Boolean;
                   User_State          : out    Traversal_State);
```

Notice that the `Phase` parameter has mode `in out`. The value passed into the call is the current phase value for the view, but the code within the provided actual can modify the value, and the underlying mechanisms will use the modified value.

One strategy for achieving this would be to store the phase value in an annotation associated with the compilation unit. Since the `Root_Element` parameter is known to be an element within the unit, you can use this to obtain the annotation and decode the value.

Note: Make sure you go to the *parent* of the `Root_Element`, so that the trick works also with incremental compilation.

♦

INFO: RDF, primary and secondary subsystems	
Applicable to:	*Fix:*
References:	*Keywords:* Design Facility, Primary, Secondary, Subsystem
Originated from: Cbh, Seh	*Revised by:*

In Volume 11 of the documentation page PM-103, it is stated that prior to copying a view from a primary subsystem into a secondary subsystem, it is necessary to promote all units in the view to the `Coded` state. This is required in order to ensure that the CDB is updated with information which is provided as a result of compilation.

The problem this causes is that the Design Facility prevents any Ada unit being promoted beyond the `Installed` state during the Preliminary and Detailed Design phases of development. This seems to imply that it is impossible to use primary/secondary subsystems during the first phases of development.

The requirement to code prior to copying a subsystem to a secondary is derived from the fact that new declarations cannot be added to a secondary subsystem. Thus the units must be coded in the primary *if they are to be coded in the secondary.*

People probably don't want to be able to only install units in the primary (the developers view), but code them in the secondary (the integration view), so it is quite OK to create secondaries from views with units in the `Installed` state.

◆

INFO: ACLs needed by Mail	
Applicable to:	*Fix:*
References:	*Keywords:* Acl, Error Log, Mail, Mailbox
Originated from: Jmk	*Revised by:*

For machines running Mail 1 *or* Mail 2, you must have the following ACLs:

```
!                                              Network_Public => R
!Machine                                       Network_Public => R
!Machine.Error_Logs                            Network_Public => RC
!Machine.Error_Logs.Mail_@                     Network_Public => RW

!Users                        Mailer or Public or Network_Public => R
!Users.@                      Mailer or Public or Network_Public => R
!Users.@.Mailbox              Mailer or Public or Network_Public => R
!Users.@.Mailbox.Main_@       Mailer or Public or Network_Public => RW
```

For Mail 1, you also need to have:

```
!Machine.Mail                          Public or Network_Public => R
!Machine.Mail.??              Mailer or Public or Network_Public => RWC
```

And for Mail 2:

```
!Machine.Transfer                              Network_Public => R
!Machine.Transfer.Distribute.??                Network_Public => RWC
!Machine.Transfer.??          Mailer or Public or Network_Public => RWC
```

♦

INFO: Mail servers	
Applicable to:	*Fix:*
References:	*Keywords:* Job, Mail, Server, Smtp
Originated from: Jmk	*Revised by:*

These jobs are needed to run Mail, and to exchange mail with other machines that are running Mail 2:

```
171   IDLE     8:48:09  Mail SMTP Carrier
214   IDLE     8:48:22  Mail LOCAL Carrier
```

These jobs are needed to exchange mail with machines which are still using Mail 1:

```
179   IDLE     8:48:17  Mail MAIL_1 Carrier
167   IDLE     8:47:20  Mail Dispatcher
210   IDLE     8:46:59  Mail Transceiver
```

This job works to keep your mailing lists and user definitions synchronized with the master copy:

```
206   IDLE     8:46:46  Mail Distribute Server
```

This job keeps track of which machines are running Mail 1 vs. Mail 2:

```
217   IDLE     8:47:50  "!Machine.Release.Mail.[_Current]".Mail_1_Monitor
```

◆

TIP: SMTP Carrier dies of inconsistencies in name maps	
Applicable to: Mail 2	*Fix:*
References:	*Keywords:* Mail, Server, Smtp
Originated from: Jls	*Revised by:*

The symptom is that mail stops getting sent or received (the message `SEND has completed` appears in the message window, but the message never actually goes anywhere). `What.Users` reveals that the SMTP carrier (which normally appears in the display as `Mail SMTP Carrier` under user `*SYSTEM`) is not running. The death is silent; no error messages are generated.

This occurs when a name defined in `!Machine.Transfer.Global` is deleted from that file, but is still referenced by `!Machine.Transfer.Local`. It happens that when trying to resolve the reference to the now non-existent name definition, the SMTP server dies.

Work-around: Delete the obsolete reference from `!Machine.Transfer.Local` and reboot the server; mail (including all messages queued during the time the server was dead) will start to flow again soon after.

Note: To avoid a repeat of the same problem, make sure to update the `Local` file whenever a name is deleted from the `Global` file. If this is too much of a pain, the only other workaround is to move all entries from the `Local` file to the `Global` file or personal name maps.

◆

INFO: Known problems in TBU	
Applicable to: Rev9, Rev10	*Fix:*
References:	*Keywords:* Target Build Utility
Originated from: Bgb, Deg, Djd, Jon, Rjg, Tls	*Revised by:*

1. `Other_Error` raised if units outside the `Units` directory

   TBU gets `Other_Error` raised in `Make_Consistent` when some of the Ada units in a subsystem to be downloaded are not found in any library in the pathname `<Subsystem>.<View>.Units`. For example, the problem occurs when some of the units can be found in `<Subsystem>.<View>.XXX`.

   Note that the PM manual (PM-24) specifically suggests the use of libraries at the same level as `Units`.

2. `Unit_Name` parameter to generic parameter procedures

   This parameter is the simple name of the unit but it should be the full Ada name. If the unit is a subunit called `A.B.C` then the simple name is just `C`. In some cases this makes it impossible to uniquely identify the unit and in other cases you have to go through additional contortions to get the full name.

3. `Host_Library` parameter to generic parameter procedures

   Page 107 of the TBU manual states that the `Host_Library` parameter "specifies the local context of the unit". But, when acting on a view, this parameter contains the view_name - which is *not* the context of the unit. The unit is, in fact, in the `.Units` directory under the view. This makes it quite difficult to construct a full pathname of the unit in question. You have to do:

   ```
   Full_Pathname :=
      Dt.Naming.Resolution(Host_Library & "?" & Unit_Name);
   ```

4. `Inconsistent_Activity` error

   When running `Move_Via_Network` or `Show_Instructions` with an activity that contains more than 1 load view it will sometimes die with an `Inconsistent_Activity` error.

5. View naming

   The `Libraries` parameter to many commands requires a list of worlds or an activity. It would be more convenient to be able to supply view names so that you don't have to create an activity just for this.

6. Current Working Directory (CWD) for network file transfer.

   When using `Move_Via_Network` to move units to the target, it moves the unit to its fully qualified pathname on the target. Some network software (like FTP running on VM) requires that you use CWD to change the context on the target and then use `Ftp.Put` with the unit's simple name.

7. Action IDs consumption in `State.Open`

   Because of action IDs consumption, there is a very finite limit to the number of times that you can call `State.Open` during any one invocation of any procedure. This has caused several problems in dealing with worlds or views with a large number of units (i.e, 200+).

8. Action IDs consumption in `Move_Via_Network`

   Also, it appears that `Move_Via_Network` uses up one action ID for each unit that gets copied over the network. Since action ID's are a limited system resource an move_via_network seems to not recycle them, when many units are moved, the system eventually runs out of them.

9. Updating ACLs

   If the ACLs are changed on a unit, the TBU treats it as if the unit itself has changed. Although this can be irritating, the worst side effect is that units are recompiled that do not necessarilty need to be.

10. Units not included in the recompilation script

    There is at least one case where the R1000 smart recompilation outsmarts the TBU. A unit will not be put in the recompilation script when all the following are true:

    - The unit needs to be recompiled on the target due to a textual change somewhere in its closure.
    - The unit does not need to be downloaded (which means the unit has not been textually changed, nor must it be changed to be Installed.)
    - At the time the `Move_Via_@` is done, the unit is in Source state.

    Work-around: Make sure that all units that can be Installed are Installed before doing a `Move_Via_@`.

11. `Go_Back_In_Time` and `Go_Back_Steps` do not update incremental file

    These procedures do not revert download times in the incremental file. This means that they do not necessarily cause files to be re-downloaded again. Additionally, if the time given for `Go_Back_In_Time` is `Beginning_Of_Time`, the incremental file is not deleted, and files will not be redownloaded.

12. `Go_Back_In_Time` sometimes will not go back to `Beginning_Of_Time`

    If `Beginning_Of_Time` is used in the `Go_Back_In_Time` command, but the number of builds has exceeded the retention count, the time for `Beginning_Of_Time` is no longer kept in the state information. This causes `Go_Back_In_Time` procedure fails with an error message stating "not enough history".

    Work-around: Rather than executing `Go_Back_In_Time`, execute `Go_Back_Steps`, using `Retention_Count - 1` as the number of steps. The retention count can be found in the state file.

13. `Other_Error` raised if the state file is frozen.

♦

INFO: Customizing the TBU	
Applicable to: Rev9_5_0 and later	*Fix:*
References: CSR 3311	*Keywords:* Customization, Subunit, Target Build Utility
Originated from: Rjg	*Revised by:*

This card lists a number of issues that are not problems in the TBU itself, but must be taken care of when customizing it.

1. TBU overwrites subunits with identical names.

   This is a case of simple pilot error that has been seen before and probably will be seen again. When a customization of TBU is created for a new target, an algorithm for naming the units on the target must be defined. The sample instantiation, `Remote_Build`, provides a good example for VAX targets. In the example, a unique serial number is appended to the unit name. If this serial number is omitted in a customization, it is likely that subunit `Foo.Y` and subunit `Mumble.Y` will both have identical target names, causing one of them to be overwritten when moved to the target.

   If a customer is seeing this symptom, have them check the naming algorithm, which is in `Compilers.Remote_Name`. `Compilers` is a package in the toolkit interface.

♦

BUG: TBU - Secondaries not moved after Add_Secondary	
Applicable to: Rev10_0_3	*Fix:*
References: CSR 3406, 3668	*Keywords:* Secondary, Target Build Utility
Originated from: Hvz, Rjg	*Revised by:*

When you add a *secondary* (previously known as *buddy*) via `Remote_Build.Add_Secondary`, the relationship between the text file and the Ada unit is set up, but no update is performed on the time information stored in the state file. This information is used to determine if the unit needs to be included during the next build, therefore the `Build` operation will not re-build the new secondary file.

This problem can be fixed on site in the `Remote_Build` source. The changes are limited to `Add_Secondary` and `Remove_Secondary` inside of:

```
!Tools.Tbu_Instantiations.Sample_Instantiation'View.Units.Toolkit_Interface.
Units'Body
```

1. Locate line 512 in `Units'Body`. It should be an *end if* statement inside of `Remove_Secondary`:

```
      if not Error.Is_Ok (Status) then
           Error.Report (Tb_Msg.Error_During_Remove_Secondary,
                         (Tb.Tools.Unit.Reference (This_Unit.Data),
                          Error.Reference (Status)));
      end if; -- Line 512.
      -- Insert code here
   end if;
```

2. Insert the following after line 512:

```
   if not Tb.Tools.Unit.Has_Secondary_Unit (This_Unit.Data) then
      -- Revert the download time for the primary unit so it, rather
      -- than the Secondary, will be moved as soon as possible.
      Tb.Tools.Unit.Download_Time
         (Data => This_Unit.Data,
          Value => Machines.Beginning_Of_Time,
          Condition => Status,
          Set_Timestamp_Too => True,
          On_Machine => This_Unit.Machine);

      if not Error.Is_Ok (Status) then
         Error.Report (Tb_Msg.Error_During_Set_Download_Time,
                           (Tb.Tools.Unit.Reference (This_Unit.Data),
                            Error.Reference (Status)));
      end if;
   end if;
```

3. Find line 481 in `Units'Body`. It should be an *end if* statement inside of `Add_Secondary`:

```
   if not Error.Is_Ok (Status) then
      Error.Report (Tb_Msg.Error_During_Add_Secondary,
                        (Parameter.Reference (Secondary),
                         Tb.Tools.Unit.Reference (This_Unit.Data),
```

```
                                Error.Reference (Status)));
        end if; -- Line 481
        -- Insert code here
    else
```

4. Insert the following after line 481:

```
        -- Revert the download time for the secondary unit so it, rather
        -- than the Primary, will be moved as soon as possible.
        Tb.Tools.Unit.Download_Time (Data => This_Unit.Data,
                                     Value => Machines.Beginning_Of_Time,
                                     Condition => Status,
                                     Set_Timestamp_Too => True,
                                     On_Machine => This_Unit.Machine);

        if not Error.Is_Ok (Status) then
           Error.Report (Tb_Msg.Error_During_Set_Download_Time,
                         (Tb.Tools.Unit.Reference (This_Unit.Data),
                          Error.Reference (Status)));
        end if;
```

5. After promoting `Units'Body`, the problem should be solved.

♦

TIP: Document Formatter - environments nested too deeply	
Applicable to:	*Fix:*
References:	*Keywords:* Document Formatter
Originated from: Unknown	*Revised by:*

The error message:

```
Formatting Environments are nested to deeply, <LINE> of <FILE>
```

occurs when there is any type of command which pushes the current environment (e.g., `~begin` something) is followed by more than 20 other environment push commands all which are not closed or ended.

This usually occurs when a pop command has inadvertently been forgotten or misplaced.

♦

PROBLEM: Document Formatter - Constraint_Error raised	
Applicable to:	*Fix:* Document Formatter Rev10_7_10
References: CSR 3682	*Keywords:* Design Facility, Document Formatter, Exception
Originated from: Vnv	*Revised by:*

There are two known cases when such a problem can occur. Both will be fixed by improving the error messages, not by supporting the corresponding configuration.

1. Text and/or lines are being placed out of page boundaries, e.g.:

```
~begin(table)
~column_definition(
heading_text=Name,
heading_mode=1,
body_mode=1,
width= 40)
~column_definition(
heading_text=Address,
heading_mode=1,
body_mode=1,
width= 50)
~column_definition(
heading_text=Page,
heading_mode=1,
body_mode=1,
width= 10)
~column_entry(Some Name)
~column_entry(Some Address)
~column_entry(~page_ref(Some Name Some Address))
~end(table)
```

   This will produce messages like:

```
::: Compose (Rev10.7.7) (Document => "<CURSOR>", Device =>
... "lineprinter", Options => "", Response => "<PROFILE>").
!!! Table wider than margins on line <LINE> of <FILE>
*** Characters beyond right edge of page ignored.
%%% Unhandled Exception Constraint_Error (Array Index).
***  2 errors detected.
!!! 1 warning issued.
++* Compose is quitting.
```

2. Problems may occur if some of your tables contain rows that will not fit on a page. The Formatter is happy to produce multi-page tables, but not multi-page rows within tables.

   Each row is automatically placed within a *keep block* so that it will not be split by a page break. Hence the error message:

```
::: Compose (Rev10.7.7) (Document => "<CURSOR>", Device =>
... "lineprinter", Options => "", Response => "<PROFILE>").
*** Keep text too large on line <LINE> of <FILE>
```

```
%%% Unhandled Exception Constraint_Error.
***  2 errors detected.
++* Compose is quitting.
```

In this case, the Formatter is forced to split the keep block across a page break because it is too big to fit on one page. So far so bad. The really nasty problem, subsequently occurs when table graphics (e.g., relative move and draw a line) are executed. Since the table row now is split across the page break, the graphics motions go off page.

In the lineprinter device, this causes a `Constraint_Error`.

In the PostScript device, the ink that misses the page is silently ignored.

Some documents will work OK in PostScript, but fail on the lineprinter. Rows that barely fit on one PostScript page will overflow a lineprinter page. This happens because the lineprinter is using a larger point size and a fixed width font.

If the problem appears when using the Rational Design Facility, it seems unlikely that you really want these big table entries anyway. The right approach may be to modify your document generators to avoid producing entries of this type.

♦

PROBLEM: Document Formatter - Constraint_Error raised	
Applicable to:	*Fix:* Document Formatter Rev10_7_10
References: CSR 3682	*Keywords:* Design Facility, Document Formatter, Exception
Originated from: Vnv	*Revised by:*

There are two known cases when such a problem can occur. Both will be fixed by improving the error messages, not by supporting the corresponding configuration.

1. Text and/or lines are being placed out of page boundaries, e.g.:

```
~begin(table)
~column_definition(
heading_text=Name,
heading_mode=1,
body_mode=1,
width= 40)
~column_definition(
heading_text=Address,
heading_mode=1,
body_mode=1,
width= 50)
~column_definition(
heading_text=Page,
heading_mode=1,
body_mode=1,
width= 10)
~column_entry(Some Name)
~column_entry(Some Address)
~column_entry(~page_ref(Some Name Some Address))
~end(table)
```

   This will produce messages like:

```
::: Compose (Rev10.7.7) (Document => "<CURSOR>", Device =>
... "lineprinter", Options => "", Response => "<PROFILE>").
!!! Table wider than margins on line <LINE> of <FILE>
*** Characters beyond right edge of page ignored.
%%% Unhandled Exception Constraint_Error (Array Index).
***  2 errors detected.
!!! 1 warning issued.
++* Compose is quitting.
```

2. Problems may occur if some of your tables contain rows that will not fit on a page. The Formatter is happy to produce multi-page tables, but not multi-page rows within tables.

   Each row is automatically placed within a *keep block* so that it will not be split by a page break. Hence the error message:

```
::: Compose (Rev10.7.7) (Document => "<CURSOR>", Device =>
... "lineprinter", Options => "", Response => "<PROFILE>").
*** Keep text too large on line <LINE> of <FILE>
```

```
%%% Unhandled Exception Constraint_Error.
***  2 errors detected.
++* Compose is quitting.
```

In this case, the Formatter is forced to split the keep block across a page break because it is too big to fit on one page. So far so bad. The really nasty problem, subsequently occurs when table graphics (e.g., relative move and draw a line) are executed. Since the table row now is split across the page break, the graphics motions go off page.

In the lineprinter device, this causes a `Constraint_Error`.

In the PostScript device, the ink that misses the page is silently ignored.

Some documents will work OK in PostScript, but fail on the lineprinter. Rows that barely fit on one PostScript page will overflow a lineprinter page. This happens because the lineprinter is using a larger point size and a fixed width font.

If the problem appears when using the Rational Design Facility, it seems unlikely that you really want these big table entries anyway. The right approach may be to modify your document generators to avoid producing entries of this type.

♦

PROBLEM: Document Formatter - line in table of contents wraps badly	
Applicable to: Document Formatter Rev10_6_8A	*Fix:*
References:	*Keywords:* Design Facility, Document Formatter, Table Of Contents
Originated from: Unknown	*Revised by:*

Entries that are too long to fit on one line in the table of contents are not formatted properly. For example:

```
------------------------------------------------------------------
                    Table of Contents

Paragraph                                                    Page

1.        THIS TITLE SHOULD FIT ON ONE LINE                    1
1.2       The Following Titles Should Not.                     2
1.2.3     This is a long title that should not fit on one line so
that we can test multi-line titles in the table of contents.   3
1.2.3.4   This is another very long title that should take up
even more room than the previous title so that we can test
multi-line titles in the table of contents.                    4
------------------------------------------------------------------
```

Fortunately, it is easy to change the `Compose` markup to avoid this problem. Near the end of the `<Subsystem>.<View>.Documentation.@_Mss` file is a line of the form:

```
~macro[toc_tab="~>~argument(text)"]
```

This line, and only this line, should be changed to:

```
~macro[toc_tab="~begin(null)~>~|~env(rm=6.0inches)~argument(text)~end(null)"]
```

After this change to its input, `Compose` will produce output like this:

```
------------------------------------------------------------------
                    Table of Contents

Paragraph                                                    Page

1.        THIS TITLE SHOULD FIT ON ONE LINE                    1
1.2       The Following Titles Should Not.                     2
1.2.3     This is a long title that should not fit on one
          line so that we can test multi-line titles in the
          table of contents.                                   3
1.2.3.4   This is another very long title that should take
          up even more room than the previous title so that
          we can test multi-line titles in the table of
          contents.                                            4
------------------------------------------------------------------
```

If you encounter this problem, the best solution is to make this change in their markup generator. If they have customized their markup generator, you might not be able to use the markup exactly as it is shown

above. For example, 6.0 is intended to be 0.5 inch less than the original right margin.
♦

PROBLEM: Document Formatter - problems in titles with commas	
Applicable to:	*Fix:*
References:	*Keywords:* Design Facility, Document Formatter
Originated from: Lss	*Revised by:*

Here's what the preview line looks like:

```
3.2  Initiation, Timing, and Sequencing
```

This is what happens when `Compose` is run:

```
::: Compose (Rev10.6.8) (Document => "<MSS_File>", Device => "POSTSCRIPT",
... Options => "", Response => "PROPAGATE,<PROFILE>").
*** Number option 'Timing' is not defined on line 100 of
... <MSS_File>.
*** Number option 'and' is not defined on line 100 of
... <MSS_File>.
*** Number option 'Sequencing()' is not defined on line
... 100 of <MSS_File>.
*** While processing toc_tab command, started on line 100
... of <MSS_File>, encountered EOF scanning for closing delimiter
... ')' on line 100 of <MSS_File>.
++* Compose is quitting.
*** Errors detected while formatting markup for document
... "<Document>".
!!! Unable to print document due to errors detected during
... document processing.
::: [Design.Print has completed].
```

The problem is that when you unwind the macro calls `~toc_tab(~paragraph_font_style(...))`, you ultimately wind up with markup that looks like:

```
~number(level=2,value=3.2,title=Initiation, Timing, and Sequencing)
```

which chokes because the formatter parses only `Initiation` as the title to the paragraph, and assumes that , `Timing, and Sequencing` are other arguments to the `~number` command.

If the argument to the title is quoted, it works:

```
~number(level=2,value=3.2,title="Initiation, Timing, and Sequencing")
```

You can also quote the original form of the markup, which then looks like this:

```
~number(level=2,value=3.2,title="~toc_tab(~paragraph_number_format(Initiation,
Timing, and Sequencing))")
```

◆

PROBLEM: Document Formatter - problems in titles

INFO: Moving MacDraw graphics to the R1000	
Applicable to:	*Fix:*
References:	*Keywords:* Document Formatter, Encapsulate, Macintosh, Postscript
Originated from: Jls, Trw	*Revised by:*

You may include in your documents graphics produced by MacDraw (or by any Mac tool that generates PostScript). To do this, include in your Markup file a `~picture` command. This command does the following:

1. Save the current virtual memory state
2. Reset all scaling
3. Translate origin to lower left corner of picture area
4. Include the PostScript as is (without any change or re-interpretation)
5. Restore virtual memory state to what it was before starting`~picture`

The `~picture` command assumes the PostScript to be included uses the *encapsulated* PostScript convention. In a few words this means that the PostScript:

1. Defines everything it uses
2. Contains no `initgraphics`, `initmatrix`, or `showpage` commands

Finally our Print Spooler checks that the first two characters of the file are `%!`; otherwise it converts the file to PostScript.

The problem is that some tools, such as MacDraw, do not produce encapsulated PostScript. You may use the procedure `Encapsulate` to encapsulate the PostScript produced by your tool.

`Encapsulate` reads the PostScript file produced by the Mac LaserWriter driver, breaks each page into a separate file and adds the definitions from the Mac's LaserPrep file that are needed to print each page. Each output file can be incorporated into a Document Formatter document using the `~picture` command.

The parameters of `Encapsulate` are as follow:

```
Encapsulate
   (File => "<CURSOR>", Root => "", First => 1, Last => Integer'Last);
```

- `File` is the name of a PostScript file generated by MacDraw. `Encapsulate` assumes that the Postscript file is either a text file or follows the MacBinary conventions. A MacBinary file has three components: A 128 byte header, which contains icon information from the Mac file; the data portion (fork) of the Mac file; and the resource portion (fork) of the Mac file. Either of the last two components may be empty. Various communication programs on the Mac support this format.
- `Root` specifies the root file name of the PostScript text files generated by `Encapsulate`. The page number of the extracted page is appended to the `Root` file name to produce the output file

name.

- Only pages in the range `First .. Last` are extracted from the file.

Note: The lower left corner of the MacDraw page becomes the lower left corner of the area allocated for the picture in the Document Formatter.

Note: When including text in the drawing, use only the built-in fonts of the LaserWriter+ (such as Helvetica, Times, Courier, Avant-Garde, etc). Do not use the Macintosh bit-mapped fonts such as those that have the name of a city (e.g. Seattle, Geneva, etc). To find out if a font is bitmapped or not, look at the PostScript file uploaded to the R1000. If it contains a contiguous block of digits many pages long, this is in all probability a bit-map

♦

PROBLEM: Document Formatter runs out of memory		
Applicable to:	*Fix:*	
References: CSR 3394, CSR 2579, CSR 3296		*Keywords:* Document Formatter, Memory
Originated from: Mac, Srp, Vnv	*Revised by:*	

While running `Compose` on a markup file the user gets a message like:

```
Ran out memory on line 13,257 of the MSS, job has 10200 cache pages
and 300 disk pages thus exceeding the 8000 page limit.
```

In some cases, the sum of cache pages and disk pages is less than page limit, and then the message doesn't make sense since it looks like:

```
Ran out memory on line 13,257 of the MSS, job has 200 cache pages
and 300 disk pages thus exceeding the 8000 page limit.
```

This can happen if the actual error is raised at very low levels; by the time it bubbles up to where the message is generated (and the values are checked) the page values have changed. In this case, see #5 below.

There are at least five different causes:

1. If there is an unmatched delimiter `Compose` buffers up all input from the begin delimiter through the end of file. If this turns out to be a very large amount of data the buffer required could cause the page limit to be exceeded.

   - For `Compose` prior to Rev10_7_7, if the page limit is exceeded before the end of file is reached then the line number given will be the last line it reached not the line containing the beginning delimiter. To find the line with the offending delimiter, turn on debugging using `~Setup(Debug)`, run until failure, see what line it was processing. Downside is `~Setup(Debug)` puts out a message for each instruction, i.e. very verbose, but one only needs to go to end of it to see where failure occured.

   - Customize the markup generator to use delimiters other than parentheses, e.g., `[]`. This is not always possible, and will not cover case where markup was generated by hand from some previous source.

2. There genuinely is not a large enough page limit to handle the job.

   Set the page limit higher by using:

   ```
   ~setup (job_page_limit = >>number<<)
   ```

   Note: Normally you can only specify up to 4 times the current default page limit as defined in your session switches. But you can fool the system and get more space by using additional `~setups`, such as:

   ```
   ~setup (job_page_limit = 32000)
   ~setup (job_page_limit = 64000)
   ```

Don't do this indiscriminately because you could run the system out of space.

3. The way bracketing commands are done can sometimes greatly affect the amount of space and time required to compose a file. Briefly, using the form:

```
~bold (>>text to be emboldened<<)
```

where there is a large amount of text to be emboldened is inefficient and it is better to use the form:

```
~begin (bold)
text to be emboldened
~end (bold)
```

4. There is a needless amount of bracket command nesting.

For large amounts of text, this can end in page limit exceeded. For example:

```
~null [
~environment (Fill = false, Justified_mode = justified)
~italic (
~program {
   >>large amount of text<<
 })]
```

In this example the `~italic`, `~program`, and `~environment` attributes are all being applied to a large amount of text; and each bracketing command (`~null`, `~italic`, and `~program`) pushes the formatting environment and buffers a copy of the large amount of text.

This can be simplified in the following way:

- `~italic` is also an option for the `~environment` command and can be added there.
- Every bracketing command effectively has a `~null` in it, so if we use the `~program` to bracket the text the `~null` can be left out.

The resulting markup looks like this:

```
~begin (program)
~environment (Fill = false, Justified_mode = justified, Italic = True)
  >>large amount of text<<
~end (program)
```

5. `Compose` makes extensive use of unbounded strings, and, given enough input, eventually stresses the `Unbounded_String` package beyond the limits of its current implementation. The symptoms are:

- The longer `Compose` runs, the slower it runs.
- The path name to the markup file is long.
- Given enough input, Compose eventually dies with a `storage error (allocation)` when trying to allocate a new unbounded string. This bubbles up and is reported as at the beginning of this card.

♦

HINT: Installing RXI on a network	
Applicable to: All versions	*Fix:*
References:	*Keywords:* Decnet, Excelan, Sun, Tcp Ip, Vax Vms, Wollongong, X Window
Originated from: Phl	*Revised by:*

This card is intended to help you fix RXI installation problems, in the case of a network installation.

The machine (if any) hosting the gateway package (Wollongong or Excelan) is hereafter called the *gateway machine*. The machine hosting the X Window server is hereafter called the *X machine*.

1. If a gateway package is involved (i.e. if you are in a VAX/VMS environment), RXI *must* run on the gateway machine. This is because RXI will need to open a Telnet connection to the R1000. Note that it doesn't matter if this machine is a "screenless" one, i.e. does not host an X Window server. To check that you are on the gateway machine, try to connect via Telnet to the R1000.

   For this reason, you should download and compile the RXI sources on the gateway machine.

   Note: This is not necessarily true on a VAXcluster.

2. Before running RXI, you need to tell it on what X machine (i.e. display) it must open the RXI window. The actual command will depend on the operating system you are using:

   - On VAX/VMS you have to type:

     ```
     $ set display/create/node=<X_Machine_Name>
     ```

     If the gateway machine and the X machine are not connected by DECNET, you will have to specify in this command the protocol used to connect to the VAXstation (DECwindow doesn't work with TCP/IP today).

   - On Sun OS, you have to define the environment variable `DISPLAY` by typing:

     - With the C-Shell (`/bin/csh`):

       ```
       setenv DISPLAY <X_Machine_Name>
       ```

     - With the Bourne Shell (`/bin/sh`):

       ```
       DISPLAY = <X_Machine_Name>
       export DISPLAY
       ```

3. It is possible that there be security problems while attempting to use RXI. These problems can stem from two sources:

   - There may be protection mechanisms in the X Window system. For instance, DECwindow provides a protection mechanism to control which machines may create windows on a particular X server. This may be changed by pulling the menu `Customize`, entry (Security). You can then specify something like `<Gateway_Machine_Name>::*` to make sure that any program running on the gateway machine will be able to create windows.

- Besides, you should also check the protection mechanisms on the gateway machine. This machine may provide ACLs and/or mask-based protections that may prevent RXI from reading certain files.

♦

INFO: Destruction utilities on machines that do not have RDF	
Applicable to:	*Fix:*
References:	*Keywords:* Design Facility, Destruction, Target Key
Originated from: Jls	*Revised by:*

The `Destruction_Utilities` supplied in the D1 release of the Software Catalog depend on the `Design_Implementation` package, which is a part of the Rational Design Facility, and therefore not on all machines.

The `Design_Implementation` package is used by the `Destruction_Utilities` in order to set the target key to NIL for PDL worlds. This is necessary in order to destroy worlds and views which contain PDL, because the CMVC operations otherwise fail.

You may however get rid of the need for `Design_Implementation`. To make the change, edit the body of `Destruction_Utilities` and remove the `with` for `Design_Implementation`. Then, change the line:

```
Design_Implementation.Set_Target (To_Value => Target_Key,
                                  For_World => The_Object);
```

to the line:

```
Compilation.Set_Target_Key (The_Key => Target_Key,
                            To_World => The_Object,
                            Response => "PERSEVERE," & Response);
```

You should now be able to compile and execute on any R1000.
♦

INFO: Destruction utilities on machines that do not have

PROBLEM: Object_Info.Is_Spec_Load_Subsystem does not work	
Applicable to: D1	*Fix:*
References:	*Keywords:* Subsystem
Originated from: Jls	*Revised by:*

Due to a change in the implementation of the `Object_Subclass` package in `!Implementation` in D1, the `Is_Spec_Load_Subsystem` function in the `Object_Info` package in the Software Catalog does not work correctly. To fix it, change the body of the function to:

```
function Is_Spec_Load_Subsystem (This_Object : in Object)
    return Boolean is
begin
    return (Object_Info.Any.Subclasses_Equal
                (This_Object,
                 Object_Subclass.Subsystem_Subclass));
end Is_Spec_Load_Subsystem;
```

♦

PROBLEM: Object_Info.Is_Spec_Load_Subsystem does

Rev. 0

TOOL: Calendar utilities - week number computation	
Applicable to:	*Fix:*
References:	*Keywords:* Calendar, Time, Week
Originated from: Mv, Pbk	*Revised by:*

```
with Calendar;
package Calendar_Utilities is

    -- This package complements Calendar and Time_Utilities with
    -- the standard week number (for non-US usage !)
    subtype Week_Number is Integer range 1 .. 53;
    function Week (Date : Calendar.Time := Calendar.Clock)
        return Week_Number;
    --
    -- returns ISO standard week number
    type Week_Format is (Short, Long, Full);
    function Week_Image (Date : Calendar.Time;
                         Format : Week_Format := Full) return String;
    function Week_Image (W : Week_Number := Week;
                         Y : Calendar.Year_Number :=
                             Calendar.Year (Calendar.Clock); -- current year !
                         Format : Week_Format := Full) return String;
    --
    --       Short:  "ww"      where ww is 2 digits of the week number
    --       Long :  "yww"       and  y the last digit of the year
    --       Full :  "Wyww"
end Calendar_Utilities;



with Time_Utilities;
package body Calendar_Utilities is

    package Tu renames Time_Utilities;
    package Cal renames Calendar;
    function "-" (Left, Right : Cal.Time) return Duration renames Cal."-";
    function "-" (Left : Cal.Time; Right : Duration) return Cal.Time
        renames Cal."-";
    function "<" (Left, Right : Cal.Time) return Boolean renames Cal."<";
    function ">=" (Left, Right : Cal.Time) return Boolean renames Cal.">=";

    function Week
        (Date : Calendar.Time:=Calendar.Clock) return Week_Number is
        This_Year : Cal.Year_Number := Cal.Year (Date);

        function Monday_Week_1 (Year : Cal.Year_Number) return Cal.Time is
            Jan_1st : Cal.Time := Cal.Time_Of (Year, 1, 1, 0.0);
        begin
            return Jan_1st -
               Tu.Day * ((Integer (Tu.Day_Of_Week (Jan_1st)) + 2)
                         mod 7 - 3);
```

```
        end Monday_Week_1;

        function Weeks_Since (Origin : Cal.Time) return Integer is
        begin
            return Integer
                       (Tu.Convert (Date - Origin).Elapsed_Days) / 7 + 1;
        end Weeks_Since;

    begin
        if Date < Monday_Week_1 (This_Year) then
            return Weeks_Since (Monday_Week_1 (This_Year - 1));
        elsif Date >= Monday_Week_1 (This_Year + 1) then
            return 1;
        else
            return Weeks_Since (Monday_Week_1 (This_Year));
        end if;
    end Week;

    function Im (I : Integer) return Character is
    begin
        return Character'Val (Character'Pos ('0') + I);
    end Im;

    function Week_Image (W : Week_Number := Week;
                         Y : Calendar.Year_Number :=
                            Calendar.Year (Calendar.Clock);
                         Format : Week_Format := Full) return String is
        Full_Form : String (1 .. 4) :=
           (1 => 'W',
            2 => Im (Y mod 10),
            3 => Im (W/10),
            4 => Im (W mod 10));
    begin
        case Format is
            when Short =>
                return Full_Form (3 .. 4);
            when Long =>
                return Full_Form (2 .. 4);
            when Full =>
                return Full_Form;
        end case;
    end Week_Image;

    function Week_Image (Date : Calendar.Time;
                         Format : Week_Format := Full) return String is
    begin
        return Week_Image (Week (Date), Calendar.Year (Date), Format);
    end Week_Image;

end Calendar_Utilities;
```

♦

TOOL: Calendar utilities - daylight saving time change	
Applicable to:	*Fix:*
References:	*Keywords:* Calendar, Mail, Time
Originated from: Gbd, Jmk	*Revised by:*

Here is a little tool that will do the daylight saving time change automatically at a specified date and time, including updating the Mail `Time_Zone` file.

You can start it up some time before the change, push it into the background, and it will do the change at the time you specify, as long as you don't reboot in the meantime. You may also put it in `!Machine.Initialize_Site`.

The invokations look like:

```
Adjust_For_Daylight_Saving_Time ("1-APR-90 02:00:00", True);
Adjust_For_Daylight_Saving_Time ("28-OCT-90 02:00:00", False);
```

And the source is as follows:

```
with Operator;
with Calendar;
with Io;
with Log;
with Profile;
with Time_Utilities;
with Transfer;
procedure Adjust_For_Daylight_Saving_Time
         (At_Time : String; Spring_Forward : Boolean) is

    Second : constant Duration := 1.0;
    Minute : constant Duration := 60 * Second;
    Hour : constant Duration := 60 * Minute;

    Time_Change_Start_Point : constant Time_Utilities.Time :=
       Time_Utilities.Value (At_Time);

    Duration_Until_Time_Change : constant Duration :=
       Calendar."-" (Time_Utilities.Convert_Time (Time_Change_Start_Point),
                     Calendar.Clock);

    Time_At_Change : constant Calendar.Time :=
       Calendar."+" (Calendar.Clock, Duration_Until_Time_Change);

    Amount_Of_Time_Change : Duration := 0.0;

    procedure Fix_Time_Zone_File (Daylight_Savings : Boolean;
                                  Name : String := "!Machine.Time_Zone") is
        File : Io.File_Type;
    begin
        Io.Open (File => File, Mode => Io.In_File, Name => Name);
        declare
```

```
            First_Line : constant String := Io.Get_Line (File);
        begin
            Io.Reset (File, Mode => Io.Out_File);
            Io.Put_Line (File, First_Line);
            Io.Put_Line (File, "Daylight_Savings => " &
                                Boolean'Image (Daylight_Savings));
        exception
            when others =>
                Io.Close (File);
                raise;
        end;
        Io.Close (File);
        Transfer.Load_Time_Zone;
    exception
        when others =>
            Log.Put_Line
               ("Failed to modify Mail time zone file (" & Name & ").",
                Profile.Error_Msg);
    end Fix_Time_Zone_File;

begin
    if Spring_Forward then
        Amount_Of_Time_Change := Hour;
    else
        Amount_Of_Time_Change := -Hour;
    end if;

    Log.Put_Line (Time_Utilities.Image
                     (Date => Time_Utilities.Convert_Time (Time_At_Change),
                      Date_Style => Time_Utilities.Day_Month_Year,
                      Time_Style => Time_Utilities.Military) &
                  " will become ...");
    Log.Put_Line (Time_Utilities.Image
                     (Date => Time_Utilities.Convert_Time
                                 (Calendar."+" (Time_At_Change,
                                                Amount_Of_Time_Change)),
                      Date_Style => Time_Utilities.Day_Month_Year,
                      Time_Style => Time_Utilities.Military));
    Log.Put_Line ("Amount of time until the time change: " &
                  Time_Utilities.Image (Duration_Until_Time_Change),
                  Profile.Auxiliary_Msg);

    delay Duration_Until_Time_Change;

    Operator.Set_System_Time
       (Time_Utilities.Image (Date => Time_Utilities.Convert_Time
                                         (Calendar."+"
                                             (Calendar.Clock,
                                              Amount_Of_Time_Change)),
                              Date_Style => Time_Utilities.Day_Month_Year,
                              Time_Style => Time_Utilities.Military));
    Fix_Time_Zone_File (Spring_Forward);
end Adjust_For_Daylight_Saving_Time;
```

♦

INFO: Kermit user's manual	
Applicable to:	*Fix:*
References:	*Keywords:* Kermit, Network, Rs232, X25
Originated from: Mam	*Revised by:*

Introduction

Kermit is a file transfer protocol which allows the transfer of text or binary files over an asynchronous port. This provides a mechanism to transfer files to remote machines via a telephone line. Kermit, on the R1000, provides a Telnet interface so that a connection to a TCP/IP host can be made via the network.

R1000 Kermit

The Kermit on the R1000 has very basic functionality. It does not have a server function. The commands that are available are: `Connect`, `Send`, `Receive`, and transfer along with `Call` and `Clear`.

Along with the `Kermit` package one must also be aware of the `Kermit_Defs` package which provides the Kermit options. With the options one can set the data port characteristics, file kind and other various parameters.

`Call`

```
procedure Call (To : String := "");
-- Initiate a Telnet connection to the given host name.
-- If the connection is successfully formed, bind it to
-- a port, and assign the port number to Kermit.Port.
```

The Telnet port which will be used is in the variable `Kermit.Port`.

`Clear`

```
procedure Clear (Port : Kermit_Defs.Port_Number := Kermit.Port);
-- If the Port is an RS-232 port, hang up the phone.
-- If it is a Telnet port, disconnect the Telnet connection.
```

`Connect`

```
procedure Connect (Port : Kermit_Defs.Port_Number := Kermit.Port;
                   Terminal_Port : Kermit_Defs.Port_Number :=
                      Kermit_Defs.Current_Port;
                   Escape : Character := Ascii.Etx;
                   Log : String := "";
                   Local_Echo : Boolean := False;
                   Options : Kermit_Defs.Physical_Options :=
                      Kermit.Options.Physical;
                   Terminal_Options : Kermit_Defs.Physical_Options :=
                      Kermit_Defs.Default.Options.Physical);
-- Carry on an interactive dialog with a remote machine via
-- the given Port.  Append a transcript to the named Log file.
-- Terminal_Port is the port for local terminal communication:
```

```
-- interaction continues until the Escape character is received
-- on the Terminal_Port.  If Local_Echo = True, input from the
-- Terminal_Port will be echoed as it is received.  Options and
-- Terminal_Options may be used to configure Port and Terminal_Port.
```

This establishes a connection to a given port. If you do a `Call` and `Connect` in the same command window then you can leave the port as the default.

The escape character by default is `Ascii.Etx`, also known as [Ctrl][C]. This can be changed if there is a possibility that it will be interpreted by the remote host. One the VAX the default escape character is `Ascii.Ls`, i.e. [Ctrl][\].

When you perform the `Connect` you will be connected to the requested port. Once you are at the given port you can begin typing to perform any setup necessary.

To talk to Kermit you type [Ctrl][C] to get it's attention then another character. [Ctrl][C]-[?] will display the following;

```
Type any one of the following characters:
^C - transmit a single  ^C
0 - (zero) transmit an ASCII NUL
B - transmit a break
C - escape the connection and return
D - disconnect the connection and return
Q - quit logging
R - resume logging
? - repeat this list of commands
```

The `Log` parameter allows you to request that a log of the Kermit session be kept. This is very useful if you want to get a file from a machine that doen't have Kermit.

Send and Receive

```
procedure Send (Source : String := "";
                Target : String := "";
                Port : Kermit_Defs.Port_Number := Kermit.Port;
                Options : Kermit_Defs.Options := Kermit.Options);
-- Transmit files via the given Port, using the Kermit packet protocol.
-- Source is the Rational pathname of the file(s) to send; wildcards
-- are OK.  Target, if non-null, is the desired remote name of the
-- file.  The default Target is to use the same name as the Source.

procedure Receive (Target : String := "";
                   Port : Kermit_Defs.Port_Number := Kermit.Port;
                   Options : Kermit_Defs.Options := Kermit.Options);
-- Receive files via the given Port, using the Kermit packet protocol.
-- Target, if non-null, is the desired local (Rational) name of the file.
-- The default Target is to use the name in the received file header.
```

Since the R1000 has no Kermit Server you must handle the transfer request at both ends.

Kermit_Defs.Options

```
type Physical_Options is
    record
        Flow_Control : Flow_Control_Type := Current;
```

```
        Parity : Parity_Type := Current;
        Speed : Bits_Per_Second := Current_Speed;
        Bits : Bits_Per_Byte := 8;
        -- Kermit will transmit more data bits if possible, but received
        -- excess bits will be discarded (set to 0).  Bits < 7 is not
        -- useful; neither Kermit packets nor alphabetic ASCII characters
        -- will fit into 6 bits.
    end record;

type Packet_Options is
    record
        Start_Of_Packet : Control_Character := Ascii.Soh;
        End_Of_Packet : Control_Character := Ascii.Cr;
        Packet_Length : Positive := 94;
        Control : Printing_Character := '#';
        Timeout : Duration := 3.0;
        Padding : Natural := 0;
        Padchar : Control_Character := Ascii.Nul;
    end record;

type Options is
    record
        Kind : File_Kind := Kermit_Defs.Text; --, binary
        Mode : File_Mode := Kermit_Defs.Create; --, overwrite, append
        Delay_Send : Duration := 3.0; -- before starting to send
        Retries : Natural := 20; -- max times to retransmit a packet
        Pause : Duration := 0.0; -- silent time between packets
        Transmit : Packet_Options;
        Receive : Packet_Options;
        Repeat : Printing_Character := '~';
        Binary : Printing_Character := 'Y';
        Block_Check : Bcc_Length := 1;
        Physical : Physical_Options;
        Debugging : Boolean := False;
    end record;

package Default is
    Packet_Options : Kermit_Defs.Packet_Options; -- pseudo-constant
    Options : Kermit_Defs.Options;               -- pseudo-constant
    -- If Options.Physical.Bits < 8, then set Options.Binary = '&'
    -- to transfer binary data.
end Default;
```

The only options that you may want to change from the default are `Kind => Kermit_Defs.Binary` to transfer a binary file (e.g. `Data` and `Index` file from `Archive`).

Transmit

```
procedure Transmit (File_Name : String := "";
                    Port : Kermit_Defs.Port_Number := Kermit.Port;
                    Log_File : String := "";
                    Options : Kermit_Defs.Physical_Options :=
                       Kermit.Options.Physical);
-- Transmit the contents of the specified file on the given Port,
-- and append them to given Log_File.  The file is transmitted
-- verbatim, without being packetized, and without error checking.
```

```
-- Options may be used to configure the Port.
```

You may use this to tranfer a file to an R1000 which has no Kermit:

1. Connect to the remote machine
2. Open a text file for editing
3. Return to your machine
4. Do the `Transmit`.
5. `Reconnect` and commit the text file
6. `Quit`.

♦

INFO: Backup tape capacity

Applicable to:	*Fix:*
References:	*Keywords:* Backup, Tape
Originated from: Dbp, Drl, Lam	*Revised by:*

```
              L x 12
Capacity =  ---------- x R
            R/D + IRG
```

Where:

```
L   = length of tape in ft.
R   = record size in bytes
D   = density of data in bytes/in
IRG = .6 in. short gap to 1.2 in. long gap  (PE)
    = .3 in. short gap to 1.2 in. long gap  (GCR)
```

Backup uses a block size of 3072 bytes. The *default* inter-record gap size is 0.3 inches.

This results in the following:

	Inches / Page	
	1600 BPI	6250 BPI
Block Size:		
1024	1.24	0.46
2048	0.94	0.31
3072	0.84	0.26
4096	0.79	0.24

So a standard backup tape has a capacity of `(2400 * 12) / 0.26`, i.e. 110,769 Kb, or 108.17 Mb.

However, this should be used only as a rough estimate since it does not include the space used by the tape labels and tape marks, nor does it include the tape that is *wasted* if the drive drops out of streaming mode.
♦

BUG: Secondary backups failure	
Applicable to:	*Fix:*
References:	*Keywords:* Backup, Corruption
Originated from: Smp	*Revised by:*

A bug has been discovered which invalidates any more than 1 secondary backup taken after a primary backup.

What works:

- A full backup followed by any number of primary backups
- A full backup, followed by any number of primary backups, followed by a single secondary backup, followed by any number of primary backups, followed by a single secondary backup, ...

What will not work:

- A full backup followed by any number of primaries, followed by multiple secondaries.

♦

INFO: Restoring from backup		
Applicable to:	*Fix:*	
References:	*Keywords:* Backup	
Originated from: Unknown	*Revised by:*	

After restoring a system from backups you *have to* do a system reboot, otherwise garbage collection will not work properly. For this reason it is more economical to boot only to EEDB. This is what happens when you boot to Kernel and then do:

```
Kernel: Go_Back_In_Time
Kernel: Defaults
```

◆

INFO: Corrupted backups	
Applicable to:	*Fix:*
References:	*Keywords:* Backup, Corruption, Exception
Originated from: Drl	*Revised by:*

When backup is saving disk information on tape, there are a number of checks to ensure the integrity of the data on the tape. Thus when customers successfully complete a full backup, they can be reasonably certain that they can restore the system to that state at some point in the future.

However, current 9-track tape technology is not 100% reliable. A customer could fail to restore from backup tapes for any of the following reasons (among others):

- Worn or damaged tapes (tape errors)
- Operator error when writing or reading the tapes
- Hardware error when writing or reading the tapes
- Backup was taken when system was over 80% full
- Trying to restore to system with fewer disks
- Losing a tape block (see below)

The point is that, due to the above, you should not rely on any one given backup as being your fallback from a catastrophic loss. Before performing work that may corrupt data, you should:

1. Take a primary backup immediately followed by a full backup. This produces two complete sets of tapes that will restore the system with no loss of data: the new full backup, and the old full backup plus the new primary.
2. Verify the last full backup running `Tape.Examine_Labels` on all tapes in the backup set (including the Blue Tape). Although it is not its intended function, this procedure can be used to verify that backup tapes are OK. If it runs successfully, then the label information is correct and the entire tape can be read without error.
3. `Archive.Save` projects and users' data.

The *missing block* syndrom

It has happened a few (five) times in the past that a backup couldn't be restored due to the exception `Block_Count_Is_Inconsistent` raised during the restore. What it means is that the number of blocks found on the tape disagrees with the tape count on the tape label. In all cases, there was only one block missing, but because we don't know where on the tape the block is missing, there's no way to fix it or recover any data from the tapes. The only solution is to take another backup or revert to the previous backup. This problem is detected by `Tape.Examine_Labels`.

The *80%* syndrom

You can't restore a backup if the kernel determines that the restore will take more that 80% of your total available space. The Blue Tape will simply hang. If you take a backup of a system that is over 80% full, this means that you will not be able to restore the backup on that system. This situation may also occur if you try to restore a backup on a system which has less disk capacity than the originating machine.

The *operator brain-damaged* syndrom

When backups are taken, the operator has a choice of how volume ids are generated. Auto generated ids are the most popular but an operator can supply their own. When, at the end of a data tape the next volume id is not present the operator is prompted to supply the next volume id before the data tape is rewound. At this point, if the tape drive is taken off-line, tape unloaded, next tape loaded, then answering the next volume id, the backup will be unrecoverable.

♦

PROBLEM: Archive.Restore fails with Layout_Error	
Applicable to:	*Fix:*
References:	*Keywords:* Archive, Exception, Layout Error, Tape
Originated from: Vnv	*Revised by:*

`Archive.Restore` fails with many messages like:

```
++* Exception !Io.Io_Exception_Layout_Error, from PC= #37B00B, #197B
   while skipping !<Directory>.<Unit>
```

and later:

```
*** In line 0: raised !Io.Io_Exceptions_Layout_Error, From PC= #37B00B,
   #197B
++* Input could not be parsed
++* Errors saving unparsed source, constraint_error (null access),
   from PC=#FD4C16, #3A0
```

You may however restore the tape by doing:

- Use `Tape.Read` to read the `Data` and `Index` files from the tape

```
Tape.Read (Volume => "<Tape_Volume_Name>",
           Directory => "<Library>",
           Options => "R1000",
           To_Operator => "Thank You",
           Response => "<PROFILE>");
```

  Important: use `Options => "R1000"` ...i.e. omit `Add_New_Line` option.

- Attempt to restore the units using:

```
Archive.Restore (Objects => "[OBJECT_1,OBJECT_2]",
                 Use_Prefix => "*",
                 For_Prefix => "*",
                 Options => "R1000",
                 Device => "<Library>",
                 Response => "<PROFILE>");
```

- Delete the `Index` and `Data` files in `<Library>`.

◆

PROBLEM: Miscellaneous problems with Archive		
Applicable to: D1, D2	*Fix:*	
References:	*Keywords:* Archive, Controlled Object, Hang-Up, Subsystem, Version, View	
Originated from: Jmk, Mat, Rjg, Vnv	*Revised by:*	

1. It appears that one or both of `Archive.Save` and `Archive.Restore` does not work for `'V(all)` (only one version is saved/restored).

2. None of the logs for any of `Archive.Save`, `Archive.List`, and `Archive.Restore` lists which versions of a unit have been saved/restored.

3. Naming used in the `Objects` param for `Archive.Copy` and `Archive.Restore` are handled inconsistently:

   ```
   Archive.Restore (Objects => "<UNIT_SIMPLE_NAME>")
   ```

   restores nothing, whereas:

   ```
   Archive.Restore (Objects => "<UNIT_SIMPLE_NAME>'Spec")
   ```

   restore the specification. On the other hand:

   ```
   Archive.Save (Objects => "<UNIT_SIMPLE_NAME>")
   ```

   saves both specification and body.

4. The documentation for `Archive.List` does *not* explain what the output consists of, specifically what the date/time value represents (is it the time object last updated, created, saved?).

5. In some cases, `Archive.Copy` and `Archive.Restore` do not make the objects controlled when copying views. One work-around is to specify a subsystem name, not a view name, as in:

   ```
   Archive.Copy (Objects => "<SUBSYSTEM>");
   ```

   Another work-around is to specify the option `Revert_Cdb`:

   ```
   Archive.Copy (Objects => "<SUBSYSTEM>.<VIEW>",
                 Options => "Revert_Cdb = True");
   ```

6. When restoring a combined view where the switch `Subsystem_Interface` was set to `True`, `Archive` thinks it is very smart to reset it to `False`.

7. `Archive.Copy`, when operating through the network, requires both machines to know their own and each others' names; that is, each machine must have entries in `!Machine.Transport_Name_Map` for itself and for the other machine. If it is not the case, the `Archive.Copy` job will hang; not error message will be produced.

♦

INFO: Using tapes to transfer files to another machine	
Applicable to:	*Fix:*
References:	*Keywords:* Ansi, Archive, Format, Tape
Originated from: Hjl	*Revised by:*

The `Format = ANSI` option on `Archive` has nothing to do with saving a group of files to be read on another machine. the LM is vague and misleading on this point.

This option is supposed to be just equivalent to doing an `Archive.Save` to a directory and them writing the `Index` and `Data` files to a tape using `Tape.Write`. It is rarely (probably never) used and should be eliminated.

◆

INFO: Using tapes to transfer files to another machine	
Applicable to:	*Fix:*
References:	*Keywords:* Ansi, Archive, Format, Tape
Originated from: Hjl	*Revised by:*

The `Format = ANSI` option on `Archive` has nothing to do with saving a group of files to be read on another machine. the LM is vague and misleading on this point.

This option is supposed to be just equivalent to doing an `Archive.Save` to a directory and them writing the `Index` and `Data` files to a tape using `Tape.Write`. It is rarely (probably never) used and should be eliminated.

♦

INFO: Archive.Save creates worlds instead of directories	
Applicable to: D2	*Fix:*
References: CSR 3304	*Keywords:* Archive, Directory, World
Originated from: Jls	*Revised by:*

`Archive.Save` used to create worlds to contain the archive when you saved to disk to a context that did not previously exist. That is, given:

```
!Users.Jls : Library (World);
  Login          : Coded Ada (Procedure_Spec);
  Login          : Coded Ada (Procedure_Body);
```

Executing:

```
Archive.Save (Objects => "login",
              Options => "R1000",
              Device => "zzz",
              Response => "<PROFILE>");
```

Used to result in:

```
!Users.Jls : Library (World);
  Login          : Coded Ada (Procedure_Spec);
  Login          : Coded Ada (Procedure_Body);
  Zzz            : Library (World);
```

But now it results in:

```
!Users.Jls : Library (World);
  Login          : Coded Ada (Procedure_Spec);
  Login          : Coded Ada (Procedure_Body);
  Zzz            : Library (Directory);
```

This change appears to be undocumented, and it may cause problems with some customers who make large archives on a regular basis, since worlds are automatically distributed around on the disks but directories are not.

Work-around: Build an archiving skin that first creates a world for the new context if it does not already exist and then calls `Archive.Save`.

♦

PROBLEM: Archive.Restore doesn't work across the network	
Applicable to: D_12_1_1	*Fix:*
References:	*Keywords:* Archive, Network
Originated from: Jgp, Jls	*Revised by:*

The D2 release note, page 34, claims that it is possible to `Archive.Restore` onto machine **A** from a tape drive attached to machine **B**. This is incorrect -- late in the release cycle (and apparently after the release note had been produced) it was discovered that network restore doesn't work.

Note: `Archive.Save` and `Archive.List` *do* work across the network. It is only `Archive.Restore` that doesn't work.

♦

PROBLEM: Archive.Restore doesn't work across the

PROBLEM: Archive.Restore doesn't work across the network	
Applicable to: D_12_1_1	*Fix:*
References:	*Keywords:* Archive, Network
Originated from: Jgp, Jls	*Revised by:*

The D2 release note, page 34, claims that it is possible to `Archive.Restore` onto machine **A** from a tape drive attached to machine **B**. This is incorrect -- late in the release cycle (and apparently after the release note had been produced) it was discovered that network restore doesn't work.

Note: `Archive.Save` and `Archive.List` *do* work across the network. It is only `Archive.Restore` that doesn't work.
♦

PROBLEM: Archive.Restore doesn't work across the

Rev. 0

INFO: Use_Prefix and For_Prefix usage with wildcards		
Applicable to:	*Fix:*	
References: CSR 3256, 3412	*Keywords:* Archive, Wildcard	
Originated from: Ams, Trw	*Revised by:*	

There is a misunderstanding of the role of parameters `For_Prefix` and `Use_Prefix` in `Archive` commands, and their behaviour with respect to wildcard characters. These have not been well documented in the past.

The `For_Prefix` is a string-matching *pattern*. The syntax is consistent with our naming expression syntax, but it is not as rich and varied as full-blown naming expressions. Only set notation (enclosed in brackets such as `[ A, B ]`) and wildcards (`#`, `@`, and `?`) are allowed. In particular, `??` is legal, but it will attempt to match two instances of `?`.

The `Use_Prefix` is a substitution string. In order to transfer any information from the `For_Prefix` to the `Use_Prefix` you must use substitution characters.

Examples:

1. Given that `!A.B` is a subsystem, `!A.C` a directory, and `!A.C.B` another subsystem, we want to copy the `Units` directories from view `Rev1_0_Spec` to view `Rev2_0_Spec` in both subsystems `!A.B` and `!A.C.B`. Trying:

```
Archive.Copy
   (Objects     => "!A.??.Rev1_0_Spec.Units",
    Use_Prefix  => "!A.@.Rev2_0_Spec.Units",
    For_Prefix  => "!A.??.Rev1_0_Spec.units",
    Options     => "",
    Response    => "<PROFILE>");
```

   will not work. A single `?` is required in `For_Prefix`. Also, the patterns must be extended to account for the full name of all possible objects. Hence the correct command:

```
Archive.Copy
   (Objects     => "!A.??.Rev1_0_Spec.Units",
    Use_Prefix  => "!A.@.Rev2_0_Spec.Units.@",
    For_Prefix  => "!A.?.Rev1_0_Spec.Units.?",
    Options     => "",
    Response    => "<PROFILE>");
```

2. Assume that `!A.B1`, `!A.B2` and `!A.B3` are three subsystems and that you want to copy units in `Rev1_Working` into `Rev2_Working`. The command:

```
Archive.Copy (Objects     => "!A.[B1, B2, B3].Rev1_Working.Units.?'C(Ada)",
              Use_Prefix  => "!A.[B1, B2, B3].Rev2_Working.Units",
              For_Prefix  => "!A.[B1, B2, B3].",
              Options     => "Replace, Changed_Objects, Ignore_Cdb",
              Response    => "<PROFILE>");
```

runs OK with `Effort_Only` added to the options. This does not work if run as shown above. Using the `@` as a wildcard in the `For_Prefix` and as a substitution character in the `Use_Prefix`, we obtain the correct command:

```
Archive.Copy (Objects     => "!A.[B1, B2, B3].Rev1_Working.Units.?'C(Ada)",
              Use_Prefix  => "!A.@.Rev2_Working.@",
              For_Prefix  => "!A.@.Rev1_Working.?",
              Options     => "Replace, Changed_Objects, Ignore_Cdb",
              Response    => "<PROFILE>");
```

◆

INFO: Tape formats we support	
Applicable to:	*Fix:*
References:	*Keywords:* Ansi, Format, Tape
Originated from: Drl	*Revised by:*

1. The tape driver currently supports 9-track tapes and Exabyte (8 mm) tapes. The tape blocks cannot be larger than 4 Kb.

2. The `Tape_Tools` package provides support for ANSI-labelled tapes level 4. That is:

   - single file / single volume
   - multiple files / single volume
   - single file / multiple volumes
   - multiple files / multiple volumes
   - all record formats (fixed, variable, segmented, also known as spanned)

   Note that the 4 Kb block restriction still applies, but is in conformance with the ANSI standard.

   The big difference among the different vendors' tape formats is what information is contained in the optional labels. We don't always process optional labels.

   - VMS-formatted labelled tapes are a variation of the ANSI standard and are generally supported (there are some restrictions).
   - IBM-standard labelled tapes could probably be supported.

3. The package `Dio` can read and write tapes. From its perspective, the tape is viewed as a very long byte stream.

4. There is a utility, built on top of `Dio` which reads and writes UNIX tar tapes. Note again, that there is a 4 Kb limit to the block size (the UNIX default is 10 Kb).

◆