The Resource Manager

Operations Manual

Research Systems
March 1990

Table of contents

Overview of the Resource Manager

Introduction

The Resource Manager is a spooling system. It accepts jobs, places them in queues depending on the resources or devices they need, and then processes those jobs as the resources become available. A job is a "work request". Each incoming job is placed in a queue according to its "attributes", e.g., a job with attributes TYPE=PRINT and PAPER=3HOLE will be placed in a print queue, and a job with TYPE=PUNCH will be placed in a punch queue.

The RM is composed of a number of processors. Each processor handles a particular kind of job request. For example, a print processor prints jobs, an execution processor executes them, and so on. Many processors have an associated device (such as a printer) to do the work required.

The RM has a command language to allow the operators and DPAs to control jobs and processors. There are commands to start and stop printers, to reroute jobs, to hold jobs, and even to change which jobs a processor will process. This manual describes the command language.

The rest of this overview describes jobs and processors in more detail.

Jobs

Jobs are the units of work for the RM. For example, a batch of cards read at a card reader is one job (an execution job), the printed output produced during execution is another (print) job, and the punched output is a third job. In fact, during execution, a batch job may produce any number of separate print and punch jobs. Each of these jobs will have its own job number. So, the job number a user gets when submitting an *batch* job is different from the job number of the printed output that batch job generates.

This is different than the HASP concept of a job. In HASP, a job was one entity that went through several steps—execution, printing, punching—in sequence. In the RM each of the units of work is considered a separate job.

Each job also has a job name which can be set by the user. A job name is an arbitrary (up to 8 character) name that always begins with a letter.

If a user does not specify a job name for a job, the RM will use a default name. For *BATCH*, *PRINT* or *PUNCH* the default name is "RM" followed by the job number. For jobs read in at card readers, the default name will be the letters "RM" followed by the number that is prepunched on the receipt card. For print and punch jobs created during execution of a batch job, the default job name is the same as the name of the execution job.

The job name is prefixed by the ccid that submitted it, in the same way that filenames are qualified. For example, SATF:TESTJOB refers to a job (or jobs) with the name TESTJOB belonging to ccid SATF.

Two jobs may have the same job name. Use JOBS or DISPLAY to list all the jobs with a given job name. When you use a job name in an RM command, *all* jobs with that name will be affected. If the name is not prefixed by the ccid, all jobs with that name belonging to any ccid are affected. If you use a job number, only that one job is affected.

When a job enters the system, it is placed in one of the RM's queues. The job itself is stored in a spool file or files. When the resources the job needs become available, it is given to one of the RM's processors to be executed, printed, or punched, etc. After the job has finished, the spool files are released (this is called purging). The RM retains the job name, job number, and the time the job was processed for another 8 hours, and can display this information via the JOBS command or systemstatus /q. Finally, the job is deleted, and no further record of it exists in the RM. Information about it is still available from *RMLOG.

For example, when an *print* job is submitted, the job is placed in a print queue, and the data to be printed is placed in a spool file. When the job's turn comes, it is given to one of the print processors. The print processor sends it to a printer. Then the job is placed in a purge queue, where it sits for four hours. It can be REPRINTed during this time. (Only print jobs are saved for four hours.) After four hours, the spool file is released.

The RM has commands to DISPLAY (or JOBS) the status of jobs, to CANCEL or RERUN them, to force a job to be processed immediately (EXECUTE, PRINT, PUNCH) or placed in HOLD, etc.

Processors and devices

A processor is an RM component which performs jobs. Each processor has a name which is used in commands to the RM. Each processor also has a distinct number, assigned when the processor is created, that is occasionally useful.

Most processors have a device that is used to do the work (these are called device processors). For example, each print processor has a printer attached.

The RM provides commands that allow you to START and DRAIN processors, INTERRUPT and RESTART after device problems, etc. The PROCESSOR and DISPLAY commands display the status of a processor.

Each processor has a definition (in file SFIL:DEFINITIONS). This definition specifies the kind of jobs the processor can handle, the device it runs, etc. For example, the definition of processor PTR6 states that it runs device PTR6, and that it accepts print jobs for a 9700 that are routed to CNTR and use 3-hole or plain paper. The RM has commands to display the definition and to make temporary changes in it (DEFINE and ALTER).

Some processors are automatically created (started) when the RM is started. The START command creates a processor according to its definition, or recreates a processor that has been destroyed (drained). Once started, a processor waits for a job to be queued for it. When a job arrives, the processor handles the job, and then, in most cases, waits for another one. This continues until someone DRAINs the processor. At this point, the processor finishes the job it is handling (if any), and is destroyed. Another START command would recreate it.

See the chapter “RM Processors” for the list of processors.

Attributes

Each job and processor has a collection of attributes that describe it. Attributes are used to determine what jobs are handled by which processor. An attribute has a left-hand-side (name) and a right-hand-side (value), e.g., PROUTE=NUBS has name PROUTE and value NUBS. A job with this attribute would be printed at NUBS. A print processor with this attribute would run a printer at NUBS (and thus print jobs with the attribute).

Every RM job has a list of attributes. These attributes describe the type of work to be done (TYPE=PRINT, TYPE=EXECUTE, ...), the priority (PRIO=LOW, ...), the route (PROUTE=UNYN, ...), the number of pages, etc. Some of these attributes are specified by the user who submits the job; others are determined by the RM. Many job attributes can be changed using the ALTER command. For example, ALTERing the value of PROUTE will reroute the job to a different site.

Every processor also has attributes. These determine what jobs the processor will handle. For example, a processor that runs a printer at NUBS will have TYPE=PRINT and PROUTE=NUBS as attributes. These attributes are set when the processor is created from the processor definition. Some processor attributes can be overridden; use the ALTER command.

Some job attributes have numeric values, e.g., PAGES=150. Some processors can be defined to accept only jobs with an attribute value in a particular range, e.g., PAGES<100.

See the chapter “Attributes” for the list of attribute names and values.

Queues

Each job awaiting processing is placed in one of the RM's queues. There is a queue for jobs awaiting execution, one for jobs awaiting printing at NUBS, another for jobs awaiting printing at CNTR, a queue for punch jobs, etc. When a processor becomes free, the RM gives it the first job with the right set of attributes from the appropriate queue. When a the job has finished processing, it goes to the purge queue and then finally to the delete queue.

For example, when the processor for the CNTR 9790 finishes a job, the RM gives it the first job from the CNTR print queue that can be printed on a 9700-type printer using plain or 3-hole paper. If a special print processor that takes 33-up label stock were running instead, the first job in the queue requiring LABEL33 paper would be selected. When the print job is finished, the RM places it in the PURGE4H queue.

Use the QUEUES command to list the jobs waiting in a particular queue.

See the chapter “RM Queues” for a list of the RM's queues.

Messages

The Resource Manager generates an enormous number of messages. These consist of informational messages about the actions it is performing (starting processors, allocating spool files, etc.), responses to requests from the operator or users, and error messages about problems detected within the RM.

Each message has an associated severity level (informational messages have lowest severity level, and error conditions highest). Messages with severity above a certain level are printed on the operator's console. The level can be set with the OPERATOR_MESSAGE command. By default error messages are printed, but not informational messages.

The RM can also vary the amount of information in a message, that is, the verbosity. The OPERATOR_MESSAGE command can also set the verbosity level for the printing of messages at the console.

RMOPR (the program used to communicate with the RM from places other than the operator's console) has its own verbosity level, set with the VERBOSITY command The default value is 5. VERBOSITY 9 causes JOB display output to include all the attributes of the job being displayed. If verbosity 10 is chosen, each response printed will include a “message header” which consists of a four-letter code for the RM component generating the message and a 7-digit message number (not of general use). The SEVERITY command sets the severity level for RMOPR.

Commands

This chapter explains the format of commands and describes common expressions in commands. It is followed by the individual command descriptions.

Entering commands

RM commands may entered at the operator's console (and via the $SY /o command) or via the program SYS:RMOPR. The command format is the same for either method, except that commands entered at the operator's console must be preceded by a $. Some commands can not be issued from the operator's console, usually because they produce large amounts of output; these restrictions are noted in the individual command descriptions.

To use RMOPR, $RUN SYS:RMOPR from any terminal. (SYS and all DSPn ccids should have access to this program. Check with a systems programmer for help with access.) RMOPR uses RM as a prefix; any RM command can be entered after this prefix. Command responses and messages are displayed on the screen.

In addition to the commands described here, any line beginning with an "*" is a comment. It is ignored by the operator command program. Comments can be used to provide documentation in a file of Resource Manager commands, such as the files that are used when starting up the RM.

Any command may be preceeded by a $-sign which is ignored. This allows commands to be entered to the program in the same manner as at the operators console.

Command format

An RM command consists of a command name and zero or more operands. The operands are separated from each other by one more blanks or commas. The description of each command includes prototypes to show the operands used with that command. For example,

DRain[**Processors**]proc-expr

A word printed in boldface is to be entered verbatim. The part of the word in capitals is required, and is the minimum abbreviation. A longer abbreviation may also be used. A word in lighter type is a "generic" item to be replaced by a specific item when entering the command. Anything enclosed in square brackets ([...]) is optional. So, in the example above, The command name is DRAIN, which may be entered as DRA, DRAI, or DRAIN. P, or PROCESSOR, etc., may be included or omitted. proc-expr is to be replaced by the name of a processor or attributes selecting some processors. Some actual commands fitting this prototype:

```
DRAIN P PTR6
dra all proute=cntr
```

Commands may be entered without regard to case. That is, DRAIN PTR6, drain PTR6, and dRaiN pTr6 mean exactly the same thing.

Common command parts

This section explains three types of expressions used in many commands:

job-expr	"job expressions" that specify one or more jobs
proc-expr	"processor expressions" that specify one or more processors
attributes	expressions used to further qualify a job or processor expression

Job-expr

A job expression, written job-expr in the command prototypes, describes one or more jobs to be acted on by the command. The general form of a job expression is

job-name-or-number [attributes]

The job-name-or-number part specifies a job number, a job name, or a set of names as described below. attributes may be used to restrict the jobs that are selected by the job-name-or-number part.

The job-name-or-number in a job-expression may be any of the following:

ccid:job-name	specifies all jobs with the specified CCID as owner and the indicated job-name. This will usually be a small number of jobs.
ccid:?	specifies all jobs whose owner is the specified CCID.
?:job-name	specifies all jobs with name job-name, regardless of the owner CCID. **CAUTION: Jobs with the same job name are not necessarily related. When using this form, be certain that you are not referring to any "extra" jobs. Never use this form to cancel a job.**
job-name	is the same as ?:job-name.
ALL or ? or *	Each of these forms means all jobs.
number	specifies the job with indicated job-number.

Any of the above job expressions can be further restricted by the addition of attributes, which restrict the jobs selected by the job-name operand to just those with the indicated attributes.

If the job-name operand is "ALL" (or ? or *) or is omitted and no attributes are given, then the attribute TYPE~=DELETE is assumed. This means that jobs in the 8-hour delete period will be ignored by RM commands that select all jobs.

If the job-name operand and the attributes are omitted, then "ALL TYPE~=DELETE" is assumed.

For example:

ALL	specifies all jobs that are not awaiting DELETE.
? TYPE=PRINT	means all print jobs.
ZZYY:? PRINT=TN	means all print=tn jobs owned by user ZZYY.
600052	means job number 600052.
XXW5:ABCDE	means all jobs owned by XXW5 with name ABCDE (including any that are awaiting DELETE).

Proc-expr

Processor expressions are used in many commands to specify a processor or processors to be acted on. They are also used in some commands to specify the job that is currently being processed by the selected processor.

The general form of a processor expression is

proc-name-or-number [attributes]

The operands of processor expressions are analogous to job expressions. The first operand selects a processor or set of processors. Attributes can be applied to restrict the selected processors to those which have certain attributes.

The forms of the proc-name-or-number operand are similar to the job-name operands:

proc-name	specifies the processors(s) with the indicated name. Names are not necessarily unique (for example, there are usually several processors with the name EXECUTE running).
proc-number	specifies the processor with given number. Processor numbers are unique.
ALL or ? or *	specifies all processors.

Use the command PROCESSORS or DISPLAY PROCESSORS to get the processor numbers for all processors with a give name.

When ALL (or ? or *) is given with no attributes, the default attributes of TYPE=PRINT OR TYPE=PUNCH OR TYPE=READER are assumed. If no proc-name or attributes are given, then ALL, with the default attributes, is assumed.

Examples:

PTR6	means the processor named PTR6.
ALL	means all PRINT, PUNCH, or READER processors.
? PROUTE=PTR6	means those processors with a PROUTE attribute of PTR6.
? TYPE~=DELETE	means all processors except the delete processor.

Attributes

Attributes expressions are used to qualify processor or job expressions to select a set of processors or jobs. They are also used in defining and starting processors, and in various other commands. A complete list of attributes appears in the chapter “Attributes”.

An attribute consists of a left-hand-side (a name), a relational operator (=, ~=, >, <, >=, <=), and a right-hand-side (value). Some attributes have symbolic right-hand-sides (i.e., words). The equal (=) and not equal (~=) operators can be used with these. For example,

```
PRIO=LOW
type~=print
```

Other attributes have numeric values. All the operators can be used with these. For example,

```
Pages<100
COPIES=5
copies~=2
```

The attribute ALL, which has no right-hand-side, means exactly what it says.

An attribute expression is a series of attributes, possible with OR operators. Attributes that are not separated by an OR are taken to be “anded”. For example,

PAPER=3HOLE PROUTE=NUBS PAGES<100

means using 3-hole paper and routed to NUBS and less than 100 pages long.

paper=3hole or proute=nubs

means using 3-hole or routed to nubs (or both). Use parentheses to indicate the range on an OR:

paper=3hole (proute=nubs or proute=unyn)

means using 3-hole paper and routed to either NUBS or UNYN.

Symbol expressions

Symbol expressions will generally be used only by systems programmers in defining the RM job attributes, queues, etc. For more information, see RMGR:STM*WP.

A symbol expression is used to indicate a particular node in the RM's tree-structured symbol table. It is an expression consisting of a sequence of symbol-symbols, connected by “.”s.

For example, TYPE.SYMBOLIC_ATTRIBUTE.INPUT is the symbol expression for the attribute TYPE=INPUT.

ALTER — change attributes of jobs or processors

Prototype

ALTer [**Job**] job-expr **TO** attributes
ALTer Processor proc-expr **TO** attributes

Operands

job-expr	specifies a job or jobs whose attributes are to be modified.
proc-expr	specifies a processor or group of processors whose attributes are to be modified.
attributes	specifies the new attributes for the job or processor.

Description

The job-expr form may not specify ALL (or ?). That is, a job name or number must be given.

Use this command to modify a job's PROUTE or PAPER.

Do not use this command to force a job to be processed immediately. Instead, use EXECUTE or PRINT or PUNCH.

You can use ALTER to modify the PRIO of a job, but the only values it may be set to are LOW, NORMAL, DEFERRED and MINIMUM (or L, N, D, M). **CAUTION: If you modify the priority of a job, the user will be charged at the rates for the new priority.**

The proc-expr form is used to change the attributes that are assumed whenever the processor is STARTed. The changes do not affect a processor that is currently running, but will will take effect when the processor(s) is next started. The START command can also specify overriding attributes. Use the DEFAULT command to restore a processor's attributes to the original RM startup values. STOPping a processor does not restore the original attributes, but STOPping the RM does.

Some processor attributes cannot be modified. In such cases, use DEFINE to create a new processor.

Examples

```
ALTER 600010 TYPE=PRINT TO PROUTE=NUBS
RMGR:RM600010 is awaiting print, P15, after 0 jobs.
```

(Use VERBOSITY 9 to have the output include the new routing.) This command will cause job 600010 to be printed at NUBS.

```
ALTER PROC PTR1 TO PRINT=TN
```

This causes PRINT=TN to be assumed any time the processor PTR1 is started subsequently.

BACKLOG — display summary of waiting jobs

Prototype **BACKlog** [**Jobs**] job-expr

Operands

job-expr — specifies the subset of jobs of interest.

Description

This command displays the backlog of selected jobs in a tabular form, along with a selection of the attributes that they have in common.

Use the QUEUES command to get a job-by-job list.

Examples

```
BACKLOG ALL PROUTE=NUBS
       3Q     75 pages DEVICETYPE=9790 PRIORITY=NORMAL PROUTE=NUBS
 1A    5Q   1036 pages PAPER=3HOLE DEVICETYPE=9790 PRIORITY=NORMAL PROUTE=NUBS
```

One line is displayed for each collection of attributes. The numbers preceding the A and Q are the number of active and queued jobs, respectively. This example shows the backlog for NUBS. The first line indicates that there are 3 normal-priority jobs using default paper waiting, none printing, with a total of 75 pages, for the 9790 at NUBS. The second line indicates that 1 job is printing, 5 are queued, for a total of 1036 pages (all 6 jobs), all requiring 3-hole paper, on the 9790 at NUBS.

```
backlog 212859
           1 Q jobs     2 pages TYPE=PRINT PROUTE=CNTR
```

The job-expr in this example specifies a single job. So, the number of pages is the size of that job.

BACKSPACE — back up a printing job a specified number of pages

Prototype

BACKSpace count [**Job**] job-expr
BACKSpace count **Processor** proc-expr

Operands

count — specifies the number of pages that the job should be backspaced. The job is actually backspaced by the next largest multiple of two, so that a two-sided job remains correctly synchronized.

job-expr — If this form is used, it must specify exactly one job, which must be active on some print processor. The attribute TYPE=PRINT is always assumed.

proc-expr — If this form is used, it must specify a print processor which is currently active. The job currently printing will be backspaced.

Description

Note that backspacing a 9700-type printer backspaces the job that is being sent to the printer, which is usually not the job being printed. Instead, REPRINT the job.

You can backspace a job up to 512 pages with this command. Successive backspaces will backup increasingly far in the print job. If a job must be backspaced more than 512 pages, REPRINT the job and then FORWARDSPACE to the desired place.

If a count greater than the number of pages already printed is given, the job will be restarted from where it began. A job which had previously been interrupted for some reason, and has started printing from somewhere in the middle, cannot be backspaced to before where it began this time. It must be REPRINTed and FORWARDSPACEd as above.

Examples

```
BACKSPACE 10 PROC PTR1
Print will be backspaced 10 pages to page number 97.
```

The job currently being printed by processor PTR1 will be backed up 10 pages to the page indicated, which is th 97th page of the job.

```
BACKSPACE 5 RMGR:TESTJOB
```

If there is a PRINT job with the name RMGR:TESTJOB active on a printer it will be backspaced 6 pages.

CANCEL — cancel a job

Prototype

CANcel [**Job**] job-expr
CANcel Processor proc-expr

Operands

job-expr — If this form is used, the jobs specified by job-expr are all cancelled. They may be either waiting (queued) or active on some processor.

proc-expr — If this form is used, it must specify a processor which is currently active. Whatever job it is processing will be cancelled.

Description

The form ALL is not allowed for the operand.

A job may be cancelled at any stage of processing, including while it is active (in execution, printing, punching).

If a job is cancelled while it is printing, the tail-sheet will still be produced.

Use this command to terminate a batch job that snarks during execution.

A job which is on the purge queue and in hold can not be cancelled. RELEASE the job first, then CANCEL it.

CAUTION: If job-expr is a job name, all jobs with that name will be cancelled. Be especially careful when using a job name or selecting jobs by attribute values.

Examples

```
CANCEL PROC PTR1
Job RM60060 cancelled.
```

This cancels the job currently being processed by PTR1.

```
CANCEL RM600010 TYPE=PRINT
```

This cancels any print jobs with the job name RM600010. Any other jobs (punch etc.) with this name will not be affected.

```
CANCEL YYCC:? TYPE=PRINT
```

This cancels *all* print jobs belonging to CCID YYCC (probably not something you should do!).

CLEAN — destory currently unused spool files

Prototype	**CLEAN** [class]	
Operands	class	is a class (1 - 7) of spool files. If this operand is specified, only unused spoolfiles in that class will be destroyed.
Description		The clean command causes all unused spool files to be destroyed. It should be used with care and usually only when things are fairly quiet, since the RM will be forced to actually create spool files until it reaches “equilibrium” again. At initialization time, *RMS does CLEAN 6 and CLEAN 7.
Note		This command is used mainly by systems programmers.

COMMAND — send command/message to Bitnet node/account

Prototype

COMmand [**WAIT**] usid@node "msg-text"
CMD [**WAIT**] usid@node "msg-text"

Operands

usid@node	is the account and node to which the command is to be sent.
msg-text	is the command or message to be sent.

Description

This command sends the specified msg-text to the indicated Bitnet account as a message or as a command to be executed at that site. If WAIT is specified, the OPR program then waits for reply messages coming back from the site and lists each. Use an attention interrupt to terminate this wait. WAIT cannot be specified from the operator's console. To see the messages returned at the console, set the message severity level to 8 with the command “OP 8”.

usid@node can also be specified as node(usid) or node.usid. If the node is omitted (just usid specified), the node at which the command is entered is assumed. *@node indicates that the message is to be sent to the node itself.

See also MESSAGE.

Examples

```
cmd Umichrly 'q system'
 From UMICHRLY: DMTCMX670I LINK WAYNEST1 CONNECT -- DMTNJI LINE 080 NOH NOD NOT
 From UMICHRLY: DMTCMX670I LINK UMICHUM CONNECT -- DMTVMB LINE 081 NOH NOD NOT
 From UMICHRLY: DMTCMX670I LINK UMDSCVM ACTIVE -- DMTNJI LINE 082 NOH NOD NOT
```

This is the status of links that are attached to UMICHRLY. The output is actually dispatches, and is displayed only if such dispatches are normally displayed. (Issuing MTS commands $set dispatches(system)=on and $set dispatches(network)=on will insure that this output does appear.)

COMMANDSTATISTICS — print RM command usage statistics

Prototype	**COMMANDStatistics** **CMDStats**
Description	This command prints the total number of times each command has been used.
Note	This command is used mainly by systems programmers.

CONTROL — issue a control command to a processor

Prototype

CONTROL [**Processor**] proc-expr **WITH** control-cmd

Operands

control-cmd is the control command.
proc-expr specifies the processors which are to receive the control command.

Description

This command may be used to send any control command to a processor. Usually control commands are sent via other commands like BACKSPACE or FORWARDSPACE. Use this command where no specific command exists.

See the section Control and Sense commands for a list of the processors accepting these commands.

Examples

```
CONTROL MERIT WITH CTL=IO_LOG=ON
No such processor.
```

This causes the MERIT processor to log CCWs and CSWs (etc.) for the bitnet link. (Use MONITOR LOGR to see the logged lines.)

Note: Currently, the response to this command is “no such processor” even when the command is successful. This is a bug; it is not known when the bug will be fixed.

Note

This command is used mainly by systems programmers.

DEFAULT — reset processor attributes to startup defaults

Prototype **DEFAULT** [**Processor**] proc-expr

Operands

proc-expr — specifies the processor or processors whose attributes are to be reset.

Description

This may be used to reset any attributes that may have been changed with the ALTER command.

If the processor is currently started, the command does not affect the attributes of the running processor. The next time it is STARTed, it will take on the default attributes.

Examples

```
DEFAULT ALL TYPE=PRINT
Processor PTR1 defaults reset.
Processor PTR2 defaults reset.
```

This resets all print processors to the initial attributes. The response includes a confirmation line for each processor.

DEFINE — define a new processor, queue, or RM symbol

Prototype

DEFine Processor proc-name[attributes]
DEFineDevice=device-name proc-name [attributes]
DEFine Queue queue-name [attributes]
DEFine SYMbol symbol-expr [number]
DEFine CCTAPES

Operands

proc-name specifies the name for the processor being defined.
device-name specifies a real device (UMMPS device name) to be acquired by the processor when it is started.
queue-name specifies the name of the queue to be defined.
attributes specifies the default attributes for the processor, or the attributes for the queue being defined.
symbol-expr specifies a node that is to be added to the RM symbol table.
number specifes a numeric value to be associated with the symbol table node defined. It defaults to 0 if not specified.
CCTAPES specifies that the carriage control tape definitions file is to be loaded.

Description

This command is normally only used from within the system definition file (SFIL:DEFINITIONS), but may be used at any time to override current definitions or add new ones.

The first two prototypes above are used to define RM processors. The first form defines a processor with no device; the second form defines a device-processor. Use the ALTER command to change the attributes of a defined processor.

The third and fourth forms are primarily for the use of systems programmers in configuring the RM. These commands are normally used only from the definitions file.

The fifth form causes the carriage control tape definitions to be processed and loaded. This command is only valid from the definitions file.

Use REMOVE to “undo” a definition.

Examples

```
DEFINE DEVICE=PTR1 PTR1 TYPE=PRINT PRINT=PN PROUTE=CNTR
```

This defines a print processor for the device PTR1 with a name of PTR1 that will process all print jobs with attributes of PRINT=PN and PROUTE=CNTR.

```
DEFINE QUEUE PRINT TYPE=PRINT
Queue PRINT created.
```

This defines a queue for any jobs with the attribute TYPE=PRINT.

```
DEFINE SYM PROUTE.SYMBOLIC_ATTRIBUTE.NEWSITE 1
```

This defines a new symbol and gives it the value 1.

Note

This command is used mainly by systems programmers.

DEVICES — display processor status

Prototype	**Devices** proc-expr	
Operands	proc-expr	specifies the processors whose status is to be displayed.
Description		This is a synonym for the PROCESSORS command. See that command for further information and examples.

DISABLE — drain processors and prevent them from restarting

Prototype **DISABLE** attributes

Operands

attributes — Specifies attributes for processors to be disabled.

Description

This command will drain all processors that match the specified attributes, and prevent starting new processors with the attributes until a corresponding ENABLE command is given.

Examples

```
DISABLE TYPE=EXECUTE
Processor EXECUTE drained.
Processor EXECUTE destroyed.
```

This drains all execution processors and prevents any more from starting. (The effect is much the same as the HASP $HOLD EX command.)

```
DISABLE TYPE=*...*
```

This will prevent additional *PRINT*, *BATCH* and *PUNCH* inputs. $HOLD *...* will have the same effect.

DISPLAY — general status display

Prototype

DISplay Jobs job-expr
DISplay Processors proc-expr
DISplay Queues [queue-expr [format-expr]]
DISplay SYMbol symbol-expr
DISplay DEFinition proc-expr
DISplay SPOOLFILES [**Job** job-expr]
DISplay NETinfo job-expr

Description

Except for DISPLAY NETINFO and DISPLAY DEFINITION, each of the above forms has a corresponding form as a separate command. For example, DISPLAY JOBS job-expr is the same as JOBS job-expr. See the descriptions of those commands for more details.

Examples

```
DISPLAY DEF PTR1
PTR1 Device=PTR1 TYPE=PRINT PROUTE=CNTR
```

```
DISPLAY JOBS SYS:PLOTRECEIPT
SYS:PLOTRECEIPT (400123) is awaiting print, P11, after 0 jobs.
SYS:PLOTRECEIPT (401398) is awaiting print, P0, held.
```

lists all jobs with job name SYS:PLOTRECEIPT.

```
DISPLAY NETINFO LFQC:?
LFQC:RM031869 (31869) from MAILERUMICHUM to MAILERUMDSCVM is class M.
LFQC:RM032518 (32518) from MAILERUMICHUM to MAILERUMDSCVM is class M.
```

DRAIN — stop processor after current job

Prototype

DRain [**Processors**] proc-expr
DRain SYStem

Operands

proc-expr	specifies the processor or processors to be drained.
SYSTEM	specifies that the whole spooling system is to be drained and then shut down.

Description

The processor(s) specified will finish the current job and then stop. If DRAIN SYSTEM is given, the RM will shut down after the last processor stops.

For the drain processor case, the specified processor(s) will finish the job(s) currently being processed and will then stop.

For the drain system case, all processors will finish the jobs currently being processed, then the Resource Manager will shut down.

To stop a processor immediately (because of a problem with the device), use INTERRUPT, RESTART, or STOP.

Examples

```
DRAIN PTR1
PTR1 drained.
```

This drains the processor with name PTR1. The response “PTR1 drained” occurs only if the processor is not active, so is able to stop immediately.

If the processor is active, the response is “PTR1 will drain”. To check later whether it has drained, PROC PTR1.

```
DRAIN ALL
```

This drains all device-processors. The attributes TYPE=PRINT OR TYPE=PUNCH OR TYPE=READER are assumed.

```
DRAIN ALL TYPE=PRINT
```

This drains all print processors.

```
DRAIN ? TYPE=PRINT PROUTE¬=CNTR
```

This drains all print processors that do not have proute CNTR.

ENABLE — allow restart of processors after disable command

Prototype **ENABLE** attributes

Operands

attributes — Specifies the attributes that are to be re-enabled.

Description

The attributes specified for ENABLE must match exactly the attributes given for a preceding DISABLE. It is not possible to enable only a subset of the processors previously disabled.

This command reverses the effect of previous DISABLE command.

Examples

```
ENABLE TYPE=EXECUTE
```

This allows creation of execution processors to be resumed, i.e., batch execution is allowed.

EXECUTE — start execution of specified job immediately

Prototype	**EXECUTE** [**Job**] job-expr	
Operands	job-expr	Specifies the job to be executed. The attribute TYPE=EXECUTE is automatically added to any attributes that are specified. The job-expr must specify a single job.

Description

The job-expr must specify a single job.

An execution processor will be started to execute the specified job. The job will begin execution immediately regardless of other conditions and attributes. That is, it will execute even if it was held or PRIO=LOW, regardless of any DISABLE commands that have been issued, and independent of the current system “maximum batch jobs” parameters.

See also RELEASE.

Examples

```
EXECUTE 205990
Processor EXECDO created, Processor number 1631.
Job W030:RM205990 is being executed.
```

EXPLAIN — display description of RM command

Prototype **EXplain** keyword

Operands

keyword — is the name of an RM command or a keyword to be explained.

Description

This command is not available directly from the operator's console.

EXPLAIN is a synonym for HELP.

Examples

```
EXPLAIN job_expr
A job_expr has the form <job_name> (<attrs>)
A job_name can be "ALL" or "?" or omitted, meaning all jobs.
It has the general form <user_id>:<name> where user_id specifies
the user owning the job, and name specifies the name of the job.
The job_name may be specified as <user_id>:? meaning all jobs
belonging to the user user_id, or ?:<name> meaning all jobs
with that name regardless of user. The job name may also be a
decimal number refering to the unique number that the Resource
Manager has assigned to the job.

Attributes may also be specified to further restrict the
selection of jobs.

For example "KHB:RM2758" or "? TYPE=PLOT"

See also attrs.
```

FILEUSAGE — display use counts for spool file classes

Prototype	**FILEusage**
Description	This is used to check file utilization, primarily to ensure that the created spool files are of appropriate sizes. The current configuration of file classes and sizes is as follows: for class c (c from 1 to 7), files contain 2**(c+1) pages, and 2**(12-c) files are allowed, except that 200 files are allowed in class 7. Unused files of classes 6 and 7 are destroyed at RM startup, and recreated as necessary. Files in other classes are not automatically destroyed.

Examples	`FILEUSAGE`
	`Spool file size class 1 has 187 uses (max 2048 thresh 1843), total usage 333723.`
	`Spool file size class 2 has 67 uses (max 2048 thresh 1843), total usage 104651.`
	`... etc.`
	max is the maximum number of spool files of that size. thresh is the number of files at which it begins issuing warning messages to the operator. uses is the number of files currently being used.

Note	This command is used mainly by systems programmers.

FORWARDSPACE — advance a print job by specified number of pages

Prototype

FORWARDspace count [**Job**] job-expr
FORWARDspace count **Processor** proc-expr

Operands

count	specifies the number of pages that the job should be forwardspaced.
job-expr	If this form is used, it must specify exactly one job, which must be active on some print processor. The attribute TYPE=PRINT is automatically assumed if it is not specified.
proc-expr	If this form is used, it must specify a print processor which is currently active. The job currently printing will be forwardspaced.

Description

You can skip ahead any number of pages within a single job with this command. If the count is greater than the remaining pages in the job, the job will be backspaced 20 pages from the end.

On a 9700-type printer this command affects the job that is being sent to the printer, which may not be the job that is actually printing.

Examples

```
FORWARD 10 PROC PTR1
```

The job currently being printed by processor PTR1 will skip ahead 10 pages.

HELP — display explanation of RM command

Prototype **Help** keyword

Operands

keyword — is the name of an RM command or a keyword to be explained.

Description

This command is not be available directly from the operator's console (i.e., $HELP does not).

This is a synonym for EXPLAIN.

Examples

```
HELP DRAIN

   DRain ("Processors") <proc_expr>
   DRain "SYStem"

The specified processors are stopped after finishing their
current jobs, if any.

The second form of the command causes an orderly shutdown of the
Resource Manager, after all the jobs currently running,
printing, punching etc., have completed. Other jobs are
prevented from starting during the interim.
```

HOLD — prevent a job from being processed

Prototype

HOLD [**Job**] job-expr
HOLD LOW
HOLD DEFERRED
HOLD MININUM
HOLD EX
HOLD *...*
HOLD Queue queue-name attrs

Operands

job-expr	specifies a job or jobs to be held.
LOW	specifies that all low priority jobs are to be held.
DEFERRED	specifies that all deferred priority jobs are to be held.
MINIMUM	specifies that all minimum priority jobs are to be held.
EX	specifies that no execute jobs are to begin execution.
...	specifies that no additional *print*, *punch* etc. may be submitted.
QUEUE	specifies that a new job queue, with name queue-name, is to be created to contain all jobs matching the specified attributes. The jobs are placed in hold.

Description

The first form (job-expr) holds the indicated job or jobs. Note that only jobs already in the RM are held -- jobs fitting the job-expr that are submitted later are not affected.

A held job will not be executed, printed, etc., until it is released with a RELEASE command, unless it is explicitly selected for processing with an EXECUTE, PRINT or PUNCH command.

The HOLD command places the job in “hold” state. A job may also be in “user-hold”. A batch job read in with DECK HOLD on the s-8 card will be put in user-hold. The output from the JOBS command will indicate if a job is in hold, user-hold or both. If a job is in both hold and user-hold then issuing a RELEASE command will return it to user-hold. A second RELEASE command will cause it to be actually released.

The form HOLD LOW causes all PRIO=LOW jobs that are not active to be placed in a special LOW queue. New low priority jobs coming into the system subsequently are also held in this queue. They remain held until RELEASE LOW is issued.

HOLD DEFERRED and HOLD MINIMUM work in the same manner as HOLD LOW.

Examples

```
hold 181904
Job DWB.:RM181904 held.
```

```
HOLD ALL TYPE=PRINT
```

This causes all PRINT jobs currently in the system to be held.

```
HOLD ALL
```

This holds all jobs except those queued for delete (the default attribute of TYPE~=DELETE is assumed).

```
HOLD LOW
```

This will hold all LOW priority jobs.

INTERRUPT — interrupt and requeue active print job

Prototype

INTerrupt [**Job**] job-expr
INTerrupt Processor proc-expr

Operands

job-expr — If this form is used, it must specify exactly one job, which must be active on some print processor. The attribute TYPE=PRINT is automatically assumed.

proc-expr — If this form is used, it must specify a print processor which is currently active. The job currently being processed will be interrupted.

Description

This command interrupts a job that is printing and puts it back in the queue *in hold*.

The processor will continue with the next job in the queue (or will stop if it was DRAINing).

The job that was interrupted may subsequently be released to resume processing. It will continue from the last checkpoint (a few pages back from the point where it was interrupted).

See RESTART and STOP for ways of dealing with device problems.

Examples

```
INT 601253
```

If job 601253 is active on a printer, it will be interrupted. *** response to this ***

JOBRANGE — set range of allowable job numbers

Prototype	**JOBRANGE** firstjob lastjob	
Operands	firstjob	specifies the first number to be used
	lastjob	specifies the last number to be used
Description		This can only be done at startup time from the initialization file. The job numbers for all jobs already in the system must fit into the new range — otherwise the RM will not process those jobs.

JOBS — display job status

Prototype **Jobs** job-expr

Operands

job-expr	specifies the job or jobs to be displayed.

Description

This command is used to find the status of a job or jobs.

At the default verbosity level, just the job name and status are displayed. If the verbosity level is 9 or greater, the job attributes and the time the job was entered are displayed.

See also QUEUES.

Examples

```
JOB ST15:LIST
Job ST15:LIST (245893) is awaiting print, P8.
```

This prints out the status of each job with job name RM24. ST15 is the user id. (245893) is the job number; it will be omitted if it is the same as the numeric part of the job name. P8 means priority (number) 8.

```
VERBOSITY 9
JOB ST15:LIST
Job ST15:LIST (245893) is awaiting print, P8.
Entered at 22:32:10 Thu Nov 26//81. TYPE=PRINT, USER=ST15,
SOURCE=RDR1, PRIORITY=LOW
```

The same, only with all the attributes.

```
JOBS ALL
```

This prints out a line for each job in the system (potentially a lot of output), except those which are done. (Done jobs are not printed because the default attributes are TYPE~=DELETE.)

LOAD — set print train and carriage control tape definition

Prototype **LOAD** printer train [cc-tape]

Operands

printer	is the name of the printer. It is the actual device name, which may not be the same as an RM processor name.
train	is the print train name
cc-tape	is the name of the carriage control tape.

Description

This command loads the specified printer with the specified character-set (print train) and carriage control tape definition. Issue this command whenever the train or carriage control definition must be reloaded; STARTing the processor does not do it automatically. After issuing the command, check the paper alignment on the printer.

The current legal train names are PN, TN, and ALA.

See PAGE:CCT_DEFS*SD for the current cc-tape names and descriptions.

This command is not used on 9700-type printers.

Examples

```
LOAD PTR1 ALA MCC33
```

This loads the train and tape for UM Library's 5.5 inch forms.

LOG_LEVEL — set severity threshold for log file messages

Prototype	**LOG_level** severity	
Operands	severity	is an integer. All messages with that severity level or higher will be written to the log file. The default is 8.
Description		See OPERATOR_MESSAGE and SEVERITY to set the threshold for console display.

MCMD — issue MTS command

Prototype	**MCmd** mts-cmd	
Operands	mts-cmd	is an MTS command to be executed. It must not be a command that causes another program to be run.
Description		This command is not be available directly from the operator's console (i.e., $MCMD something does not work). The specified command is executed, with control returning to the command program afterwards. An mts command cannot be issued from within the program by just preceding it with a $-sign, because the $-sign is required for RM commands from the console.
Examples	`MCMD DISPLAY VMSIZE`	

MESSAGE — send command/message to Bitnet node/account

Prototype

MESsage usid@node msg-txt
MSG usid@node msg-txt

Operands

usid@node	is the account (usid) and location (node) to which the message is to be sent
msg-txt	message text, enclosed in primes or quotes (").

Description

This command sends the specified messagetext to the indicated Bitnet account.

usid@node may be replaced by node(usid) or node.usid. If node is omitted (usid used alone) the node at which the command is entered is assumed. *@node means to send the message to the node itself.

This command is the same as COMMAND (or CMD), except that there is no WAIT parameter.

MONITOR — display RM messages as they occur

Prototype

MONitor LOGR
MONitor LOGR msg#
MONitor LOGR msg#-msg#
MONitor LOGR subj=value

Operands

msg#	only messages with number msg# are displayed.
msg#-msg#	only messages with numbers in the range msg# to msg# are displayed.
subj=value	This resticts monitoring to messages which contain the specified subject as one of the message operands, and with the value of the subject matching value. The match between the value of the message operands and the string value is done case independent. “?” may be used to match arbitrary characters.

Description

This command causes a continous display of all RM messages above the severity level selected by the SEVERITY command. The display continues until interrupted by an attention interrupt.

MONITOR LOGR (with no arguments) causes all messages to be displayed.

CAUTION: MONITOR should generally be used only when absolutely necessary since it can seriously degrade system performance. If the terminal used to display the output cannot keep up, the RM will come to a screeching halt. In particular, make sure the terminal does not pause.

The messages and message numbers are listed in macros RM_MESSAGES and DSP_MESSAGES in COPY:RMGR*SQL. Actual message text is in RMGR:EFMSA and DSP:EFMSA.

Examples

```
monitor logr 1000000-1999999
```
displays only messages from the “DSP layer”.

```
monitor logr procname=staff
```
displays only messages containing procname=staff as a message operand.

Note

This command is used mainly by systems programmers.

MTS — return to MTS command mode

Prototype **MTS**

Description

This command can only be used when running the operator command program from an MTS terminal. It will produce an error message if entered from the operator's console.

This command returns to MTS. The operator command program may be subsequently resumed with the MTS $RESTART command, as long as no other program has been run.

OPERATOR_MESSAGE — set severity of messages printed at console

Prototype **OPerator_message** [severity [verbosity]]

Operands

severity — is a number from 0 to 20. It defaults to 9. Only messages with this severity or above will be printed.

verbosity — is a number from 0 to 10. It defaults to 10. This sets the verbosity level for those messages that are printed.

Description

The parameters set determine the messages that get printed at the operator's console. If the command is issued with no operands, the severity and verbosity are reset to the default values.

The defaults are set up so that only error messages appear.

See also SEVERITY and VERBOSITY to set these thresholds when using SYS:RMOPR, and LOG_MESSAGE to set them for logging.

Examples

```
OP 8 9
```

This sets the severity to 8, and the verbosity to 9.

PRINT — print particular job immediately

Prototype

PRINT [**Job**] job-expr **ON** proc-name

Operands

job-expr — specifies the job to be printed. The attribute TYPE=PRINT is automatically added to any attributes that are specified.

proc-name — specifies the processor which is to print the job. It will be automatically started with the attributes necessary to print the job.

Description

The job-expr must specify a single job.

The processor must be drained.

The print processor will be started to print the specified job. The job will be printed regardless of other conditions and attributes. That is, it will print even if it was held or PRIO=LOW, and regardless of any DISABLE commands that have been issued. The processor will be automatically drained when the job is completed.

Use REPRINT to print another copy of a job that has already been printed.

Examples

```
PRINT JUDE:ACCTNG1 ON PTR1
Processor PTR1 created, Processor number 364.
Job JUDE:ACCTNG1 is being printed on PTR1.
```

This will start printing the job using processor PTR1.

PROCESSORS — display processor status

Prototype **Processors** [proc-expr]

Operands

proc-expr specifies the processors whose status is to be displayed.

Description

This command is used to determine the status of an RM processor. In normal operation the processor should either be active or waiting for work.

if the operand is omitted, ALL is assumed. ALL means all “real device” processors.

Examples

```
P PTR1
Processor PTR1  (34) is waiting for work.
          TYPE=PRINT PROUTE=CNTR.
```

The message describes the current status and the attributes with which the processor was started. 34 is the processor number.

```
proc ptr6
No such processor.
```

No processor with name PTR6 is currently active. This does not imply that one is not defined, i.e., START PTR6 might work.

PUNCH — start punching particular job

Prototype	**PUNCH** [**Job**] job-expr **ON** proc-name	
Operands	job-expr	Specifies the job to be punched. The attribute TYPE=PUNCH is automatically added to any search attributes that are specified.
	proc-name	specifies the processor which is to punch the job. It will be automatically started with the attributes necessary to punch the job.
Description		The job-expr must specify a single job. The processor specified must be drained. The punch processor will be started to punch the specified job. The job will be punched regardless of other conditions and attributes. That is, it will be punched even if it was held or PRIO=LOW, and regardless of any DISABLE commands that have been issued. The processor will be drained when the job has completed. Use REPUNCH if the job has already started to punch.
Examples	`PUNCH LCIR:CIRDLY1 ON PCH2`	This will start punching the specified job using the processor PCH2.

QUEUES — display status of RM job queues and jobs

Prototype

Queues [queue-expr [**WITH** [format-expr]]

Operands

queue-expr Specifies the queue or queues to be displayed by name (ALL may be used to display all queues) and optionally a set of attributes to be used in selecting the jobs on the queue to display.

format-expr This consists of a list of attribute left hand side keywords. The values of these attributes will be printed for each job.

Description

This command displays the number of jobs in each of the selected job queues and optionally lists out selected jobs on them. For each queue, the values of attributes specified by format-expr are printed in the order in which they appear in format-expr. Only the values are printed; the attribute names are not.

To list the individual jobs in a queue, include a format-expr in the command.

See also JOBS, BACKLOG.

Examples

```
QUEUE CNTR WITH PAGES
Queue CNTR            has   3 jobs TYPE=PRINT PROUTE=CNTR.
   Job W04K:RM094266 (94266) is awaiting print, posn 0.              1
   Job W18X:LISTING  (94270) is awaiting print, posn 1.              7
   Job W043:RM095505 (95505) is awaiting print, posn 2.             12
```

The number in parentheses is the job number. The number at the far right is the number of pages in the job.

The reply gives the number of jobs in the queue, and the full set of attributes for the queue, followed by a list of jobs if job selecting attributes were given.

```
QUEUE
Queue MINIMUM  (held) has   23 jobs TYPE=EXECUTE PRIORITY=MINIMUM.
Queue MINIMUM  (held) has    3 jobs TYPE=EXECUTE PRIORITY=MINIMUM.
Queue DEFERRED (held) has   17 jobs TYPE=EXECUTE PRIORITY=DEFERRED.
Queue NUBS            has    2 jobs TYPE=PRINT PROUTE=NUBS.
Queue IMPORT          has  312 jobs TYPE=FILE LINK=LOCAL.
Queue MESSAGE         has    0 jobs TYPE=MESSAGE LINK=LOCAL.
Queue DELETE          has   67 jobs TYPE=DELETE.
Queue PURGE4H         has   33 jobs TYPE=PURGE ITYPE=PRINT OR
                                    TYPE=PURGE ITYPE=RAW.
```

RELEASE — allow processing of held jobs

Prototype

RELease [**Job**] job-expr
RELease LOW
RELease DEFERRED
RELease MINIMUM
RELease EX
RELease *...*
RELease Queue queue-name

Operands

job-expr	specifies a job or jobs to be released.
LOW	specifies that all low priority jobs are to be
DEFERRED	specifies that all minimum priority jobs are to be
MINIMUM	specifies that all deferred priority jobs are to be released.
EX	specifies that execute jobs may begin execution.
...	allows *print*, *punch*, and *batch* jobs to be submitted.
queue-name	destroys the specified queue (but not the jobs on the queue). The jobs on the queue will be moved to other queues.

Description

The first form releases the indicated job or jobs. The jobs may be either be in hold as a result of a previous HOLD command, or in user-hold. RELEASE will remove either type of hold. If a job is in both hold and user-hold then issuing a RELEASE command will return it to user-hold. A second RELEASE command will cause it to be actually released.

The form RELEASE LOW causes all PRIO=LOW jobs to be released for processing. New low priority jobs coming into the system subsequently will not be held. A later HOLD LOW command will cause any low priority jobs that have not been processed to be returned to hold. DEFERRED and LOW are similar.

The RELEASE JOB form will *not* release a low-priority job that is held by a HOLD LOW (and similarly for deferred- and minimum-priority jobs). Note, however that the EXECUTE, PRINT or PUNCH commands can be used to force processing such a job.

Examples

```
release dwb:?
Job DWB.:RM181904 is not held.
Job DWB.:RM053563 (53564) is not held.
Job DWB.:RM024572 (24572) released.
Job DWB.:RM024572 (24572) released.
Job DWB.:RM024607 (24607) released.
Job DWB.:RM9881 (200112) released.
```

A line is printed for each job matching the job-expr, stating whether it was released.

```
release low
Queue LOW destroyed.
12 jobs requeued.
```

This releases all LOW priority jobs.

REMOVE — undefine a processor, queue, or symbol

Prototype

REMove Processor proc-expr
REMove Queue queue-name
REMove SYMbol symbol-expr

Operands

proc-expr specifies a processor or processors to be undefined.
queue-name specifies a queue or queues to be removed. Some queues are permanent and cannot be removed. Currently the DELETE queue and the HOLD queue fall into this catagory.
symbol-expr specifies a symbol or symbols to be removed from the RM symbol table.

Description

This command is the opposite of DEFINE; it removes the definition.

This is *not* the way to STOP a processor or RELEASE a held queue.

Examples

```
REMOVE PROC PTR1
Processor definition for PTR1 removed.
```

Note

This command is used mainly by systems programmers.

REPRINT — print a job again

Prototype

REPRINT [**Job**] job-expr
REPRINT Processor proc-expr

Operands

job-expr — If this form is used, it must specify exactly one job. The job will be reprinted from the beginning. The attribute TYPE=PRINT is automatically appended to the search expression.

proc-expr — If this form is used, it must specify a print processor that is currently active. The job currently printing will be restarted from the beginning.

Description

A job can be reprinted for up to four hours after it has been sent to the printer.

PRINT will force immediate printing of a job that has not already been printed.

Examples

```
REPRINT PROC PTR1
*** *** is there a response to this?
```

The job currently being printed by processor PTR1 will be started again from the beginning.

```
REPRINT 157820
Job RMGR:RM157820 will be reprinted.
```

If job 157820 is active on a printer it will be restarted; if the job is not active, it is queued so that it will be reprinted. The job will be reprinted from the beginning.

REPUNCH — punch a job again

Prototype

REPUNCH [**Job**] job-expr
REPUNCH Processor proc-expr

Operands

job-expr — If this form is used, it must specify exactly one job, which must be active on some punch processor. The attribute TYPE=PUNCH is automatically appended to the job-expr.

proc-expr — If this form is used, it must specify a punch processor which is currently active. The job currently being punched will be restarted from the beginning.

Description

You can start a job punching again from the beginning with this command, as long as it has not completed punching. (Punch jobs are purged immediately -- they are not held for four hours as print jobs are.)

Use PUNCH to force punching of a job that has not yet started to punch.

Examples

```
REPUNCH PROC PCH2
```

The job currently being punched by processor PCH2 will be started again from the beginning.

RERUN — stop executing job and rerun later

Prototype	**RERUN** [**Job**] job-expr	
Operands	job-expr	This must specify exactly one job, which should be in execution. The command has no effect on execute jobs that are not currently executing. The attribute TYPE=EXECUTE is automatically appended to the job-expr.
Description		The job is stopped immediately, and returned to the execution queue *in hold*. It must be subsequently released with the RELEASE command. When released, the job will be rerun from the beginning.
Examples	`RERUN DWB:RM600100` `Job DWB:RM600100 will be rerun.`	Job DWB.:RM600100 is returned to the execution queue.

RESTART — stop and restart a processor

Prototype	**REStart** [**Processor**] proc-expr	
Operands	proc-expr	specifies the processor or processors to be restarted.
Description		RESTART STOPs and then STARTs the processor. When a PRINT processor is restarted, the job currently printing is resumed from the last checkpoint (a few pages back). This is NOT like the HASP RESTART command for printers. Use the REPRINT command to begin printing the current job again. This command is intended for problem recovery, and should be used with discretion. On a 9700-type printer, user output may be adversely affected.

Examples

```
RESTART PTR1
Processor PTR1 restarted.
```

SENSE — issue sense command to a processor

Prototype

SENSE Processor proc-expr **WITH** sense-cmd

Operands

sense-cmd is the sense command.
proc-expr specifies the processors which are to receive the sense command.

Description

This command may be used to send any sense command to a processor. All processors accept the commands JOB, DEVICE, NODE, STATUS and TASK. Certain processors may support other sense commands.

See the section Control and Sense Commands for a list of the processors accepting these commands.

Examples

```
SENSE BITMSG WITH MSGSTATUS
Processor BITMSG has export jobs enabled, and import jobs enabled.
Currently, the exporter subling is idle,
and the importer subling is idle.
```

```
SENSE ARGUS4 WITH TASK
Processor ARGUS4 has auxiliary task# 252.
```

This displays the task number of the processor, which is useful when the printer gets “stuck”.

Note

This command is used mainly by systems programmers.

SET — set available tape drives and array processors

Prototype

SET TAPES = number
SET TAPEDRIVES = number
SET APS = number
SET ARRAY_processors = number

Operands

number is the number of tapedrives or array processors permitted.

Description

The first two forms set the number of tapedrives that are available for allocation to batch jobs that specify TAPES=n on the signon command. The second two forms set the number of array processors that are available for allocate to batch jobs that specify APS on the signon command. Note that UM has no array processors.

*RMS does SET ARRAY_PROCESSORS = 0 and SET TAPEDRIVES = 18.

SEVERITY — set which messages are displayed

Prototype	**SEVERity** severity	
Operands	severity	a number between 0 and 20 (inclusive).

Description

This command sets the “severity-level” that determines which RM messages are displayed. If issued from the operator's console, It has the same effect as the severity operand of the OPERATOR_MESSAGE command. See also VERBOSITY, MONITOR.

LOG_MESSAGE sets the severity for messages written to the log file.

SINGLESPACE — ignore carriage control for printing job

Prototype

SINGLEspace [**Job**] job-expr
SINGLEspace Processor proc-expr

Operands

job-expr — If this form is used, it must specify exactly one job, which must be active on some print processor. The attribute TYPE=PRINT is automatically appended.

proc-expr — If this form is used, it must specify a print processor which is currently active. The job currently printing will be singlespaced.

Description

If a job is wasting paper by continuous page ejects, this command can be used to cause it to singlespace each line. Check first that the user is not doing the ejects intentionally.

This command is useful mainly for line printers. On a 9700-type printer, SINGLESPACE will affect the job being sent to the printer, which is usually not the job being printed.

Examples

```
SINGLE PR PTR1
```

The job currently being printed by processor PTR1 will be singlespaced for the remainder of its output.

SPOOLFILELIMIT — increase number of spoolfiles allowed

Prototype	**SPOOLFILELIMIT** class limit	
Operands	class	is the number (1 - 7) of a spoolfile class.
	limit	is the allowed number of files. The maximum is 4096.
Description		The spoolfilelimit command is used to increase the maximum number of spoolfiles allowed in a given class. The increased limit is only effective during the current invocation of the RM. the next time the RM is started, the limit reverts back to its 'usual' value. See the FILEUSAGE command for a list of the “usual” limits for each class of spool file.
Note		This command is used mainly by systems programmers.

SPOOLFILES — display spool files used by a job

Prototype **SPOOLFILES** [**Job**] job-expr

Operands

job-expr — Specifies a job whose spool-file names are to be printed out.

Description

This is the same as the command DISPLAY SPOOLFILES.

Each job is stored in its own spool file or files.

Examples

```
spoolfiles 204661
W58G:RM204661 is awaiting print, P13, after 0 jobs.
File 1  SFILSPOOLFIL00C3
```

The output lists the spool files the job is using.

Note

This command is used mainly by systems programmers.

START — start processor, or the RM

Prototype

STart SYStem
STart SYStem [prefix] [**WARM**]
STart SYStem [prefix] [**COLD**]
STart [**Processor**] proc-expr [**WITH** attributes]

Operands

SYSTEM	specifes that the whole RM is to be started.
prefix	is a ccid containing the spool files and checkpoint information. This is used only when running test versions of the RM.
WARM	specifies a warm start is to be performed. This is the default.
COLD	specifies a COLD start is to be performed. The RM will be reinitialized. **CAUTION: All jobs currently in the system will be discarded. Be absolutely certain that you are doing the right thing before entering this command.**
proc-expr	specifies a processor or set of processors to be started.
attributes	specifies attributes to be given the processors that are started.

Description

The WITH attributes part is allowed only if the proc-expr specifies a single device.

The START SYSTEM form is used in starting up the whole Resource Manager. It is normally done as part of the IPL process.

The options “prefix” and “COLD” should be used only by systems programmers. The prefix defines a ccid other than SFIL for use by the RM, and COLD causes all queued jobs to be lost and the RM to start with a clean slate.

The START PROCESSOR form is the normal way to start print, punch and reader processors as necessary during operation. Attributes may be specified to override the defaults that were given when the processor was defined, or set with the ALTER command. Additional attributes (for which no default is supplied) may also be added at the time the processor is started.

Examples

```
start ptr2
Processor PTR2 created, Processor number 1827.
```

This starts PTR2.

```
START ALL TYPE=PRINT PROUTE=CNTR
```

This starts all CNTR print processors with the default attributes.

```
START ALL
```

This starts *all* device processors with their default attributes.

STATUS — display general status information

Prototype **STATus**

Description

The information given indicates the current spool file utilization, the number of queued jobs and active jobs, and what RM functions have been disabled.

Use BACKLOG and QUEUES to display more specific information.

Examples

```
STATUS
Resource Manager is initialized. 25 queued jobs, 5 active jobs.
Spool space utilization is 18% with 756 free spool files.
Processors TYPE=EXECUTE are disabled.
```

STOP — stop a processor, the operator program, or the RM

Prototype

STOP
STOP SYStem
STOP [**Processor**] proc-expr

Operands

SYSTEM	specifies that the whole RM is to be shut down immediately.
proc-expr	specifies a processor or processor to be stopped.

Description

The first form (with no parameters) is used to terminate the console program when running it from an MTS terminal. An end-of-file has the same effect.

The second form is used to shutdown the whole RM. All processors will be stopped. This should only be used as a result of serious problems within the RM (use DRAIN SYSTEM otherwise). It is possible that some of the tasks involved might not go away, in which case the STOP or BLAST job-requests must be used. See the section on RM problem handling.

The third form is used to stop a processor immediately. This should generally only be used in case of problems with a device. The active job will be requeued, and printed later beginning where it stopped (not necessarily from the beginning). DRAIN should be used at other times, so that the active job, if any, is allowed to finish.

See also RESTART, INTERRUPT.

Examples

```
STOP PTR1
Processor PTR1 destroyed.
```

SUMMARY — display amount of work in the system

Prototype	**SUMmary** job-expr	
Operands	job-expr	specifies the jobs to summarize.
Description		This command can be used to find the total number of pages to print or cards to punch by a specfied group of jobs. See also BACKLOG, QUEUES.
Examples	`SUMMARY ALL TYPE=PRINT PROUTE=CNTR` `3 active jobs, 4 queued jobs, representing 32 pages.`	

SWITCHLOG — change RM log file

Prototype	**SWITCHlog**
Description	The RM logger task cycles through a pool of three log files. After it fills a file, it empties and then reuses the oldest log file. Each file holds about an hour or two worth of data. The currently active log file is always locked while the RM is active. This command can be used to force the logger to switch to the next file so that the data logged can be examined. Use RMGR:PRINTLOG to print the log in a readable format.
Examples	`SWITCHLOG` `Switching log files from "SFILLOGFILE1" to "SFILLOGFILE2".`
Note	This command is used mainly by systems programmers.

SYMBOL — display RM symbol definition

Prototype **SYMbol** symbol-expr

Operands symbol-expr is a node in the RM symbol-table.

Description This prints out the value of each of the nodes immediately below specified symbol-table node. See the symbol table writeup in RMGR:STM*WP for more information.

Examples

```
SYM PAPER.SYMBOLIC_ATTRIBUTE
Subproperty "LABEL33" has value 0.
Subproperty "3HOLE" has value 0.
Subproperty "ANYSTANDARD" has value 0.
Subproperty "LABEL24" has value 0.
Subproperty "SP1" has value 0.
Subproperty "SP2" has value 0.
Subproperty "PLAIN" has value 0.
Subproperty "LIBCIRC" has value 0.
Subproperty "LIBRECALL" has value 0.
```

Note This command is used mainly by systems programmers.

TAPEDRIVES — set available tape drives

Prototype	**TAPEDRIVES** [number]	
Operands	number	is the number of tape drives that may be used. If it is omitted, the number is reset to 0.
Description		This command sets the number of tapedrives that are available for use by batch jobs to be scheduled by the RM. *RMS sets the number of tape drives available to 18. This command is a synonym for “SET TAPES=n”.

TRACE — turn CLParser tracing of commands on or off

Prototype	**TRACE ON** **TRACE OFF** **TRACE FULL**	
Operands	ON FULL OFF	enables tracing of semantic actions performed. enables full syntax and semantics tracing. disables tracing.
Description		This is used only to debug the RM command language.
Note		This command is used mainly by systems programmers.

VERBOSITY — set verbosity of displayed messages

Prototype **VERBosity** [number]

Operands

number	is a number between 0 and 10. The default is 5.

Description

This command controls the “verbosity” of printed information. A verbosity of 0 implies print nothing while a verbosity of 10 implies print everything. When the verbosity is set to 10, a message header is included. Many messages have only verbosity level 1 at present, so setting the verbosity to any value between 1 and 10 will have no effect on the message text for these messages. Note that the default verbosity is such that the JOBS display does not include the attributes of the job. There are currently three effective verbosity levels -- 0, 9 and 10. VERBOSITY 9 will cause attribures to be displayed. VERBOSITY 10 adds message numbers and module ids.

When issued from the operator's console, this command has the same effect as the verbosity operand of the OPERATOR_MESSAGE command.

Examples

```
verb 5
jobs 181904
DWB.:RM181904 is awaiting print, P11, after 0 jobs.


verb 9
jobs 181904
DWB.:RM181904 is awaiting print, P11, after 0 jobs.  Entered at 00:06:04 Sun Nov
05/89.
       TYPE=PRINT DELIVERY=ARGUS2 DEVICETYPE=LINE HOST=UM PRIORITY=MINIMUM
PROUTE=CNTR PAGES=5 PROJ=WDWB PROC=PR834 SOURCE="DS0B" USER=DWB. USERNAME="Don
Boettner".


verb 10
jobs 181904
CIJ .2012002 DWB.:RM181904 is awaiting print, P11, after 0 jobs.  Entered at
00:06:04 Sun Nov 05/89.
       TYPE=PRINT DELIVERY=ARGUS2 DEVICETYPE=LINE HOST=UM PRIORITY=MINIMUM
PROUTE=CNTR PAGES=5 PROJ=WDWB PROC=PR834 SOURCE="DS0B" USER=DWB. USERNAME="Don
Boettner".
```

List of RM queues

Every RM job that is not active is on a queue. The queues currently used on UM are:

Queue	Contents
MINIMUM	minimum priority batch jobs
DEFERRED	deferred priority batch jobs
LOW	low priority batch jobs
UNYN	print jobs for UNYN
NUBS	print jobs for NUBS
CNTR	print jobs for CNTR
RPRINT	print jobs for all “local printers”
ARGUS	print jobs for ARGI
EXECUTE	execution jobs (batch)
EXPORT	print, punch, execution jobs to be sent to other hosts
PUNCH	punch jobs
UMIPHYS	jobs to be transmitted to UMIPHYS bitnet site
MERIT	jobs to be transmitted to MERIT bitnet site
UMICHUB	jobs to be transmitted to UMICHUB bitnet site
UMICHRLY	jobs to be transmitted to UMICHRLY bitnet site
IMPORT	incoming Bitnet files
MESSAGE	incoming Bitnet messages
ACC	accounting reconciliation
PURGE4H	print jobs to be purged (held for 4 hours)
NETPURGE	Bitnet jobs to be purged (other than those from MESSAGE queue)
MSGPURGE	jobs from MESSAGE queue to be purged
PURGE	all other jobs to be purged
DELETE	jobs to be deleted (held for 8 hours)
HOLD	all jobs being held

List of RM processors

The processors that are currently defined are listed below. These will not necessarily all be started at any given time. Use the DISPLAY DEFINITION command to verify the listings. If you discover that the list needs updating, please see that it gets done!

Processor	Device	Description
execute		execution queue
fastexec		execution queue, small jobs (*** *** <20sec *** ***)
bitmsg		bitnet messages
purge		purge queue for execute and punch jobs
purge4h		purge queue for print jobs (4 hour hold)
netpurge		purge queue for network jobs (4 hour hold)
msgpurge		purge queue for message jobs (4 hour hold)
rm2.pr1	printer	FLNT line printer, tn
rm2.rd1	reader	at FLNT
rm6.pr1	printer	at SOPH
rm6.pu1	punch	at SOPH
rm6.rd1	reader	at SOPH
rm10.pr1	printer	at FADM
rm10.rd1	reader	at FADM
rm14.pr1	printer	at MDPH
rm14.rd1	reader	at MDPH
rm19.pr1	printer	at FOrd
rm19.pu1	punch	at FOrd
rm19.rd1	reader	at FORD
rm23.pr1	printer	MLRC printer, tn
rm23.rd1	reader	at MLRC
rm35.pr1	printer	MVEL printer, tn
rm35.pr2	printer	MVEL printer, tn
rm35.pu1	punch	at MVEL
rm35.rd1	reader	at MVEL
rm41.pr1	printer	MEDS printer, tn
rm41.pu1	pnch	at MEDS
rm41.rd1	reader	at MEDS
rm99.pr1	printer	testing
rm99.pu1	punch	testing
rm99.rd1	reader	testing
ptr1	PTR1	CNTR line printer
ptr1lib	PTR1	CNTR line printer, ala train, cctape mcc33, libcirc (5.5inch high) paper
ptr2	PTR2	ARGI Xerox 4050, 3-hole or plain paper
ptr6	PTR6	CNTR Xerox 9790, 3-hole or plain paper
holesin	PTR6	CNTR Xerox 9790, paper set for onesided 3-hole
holesout	PTR6	CNTR Xerox 9790, paper set for twosided 3-hole
ptr7	PTR7	NUBS Xerox 9790, 3-hole or plain paper
ptr8	PTR8	UNYN line printer
ptr9	PTR9	UNYN Xerox 9790, 3-hole or plain paper

argus1		ARG1 local printer
argus2		ARG2 local printer
argus4		ARG4 localprinter
arox		AROX local printer
chur		CHUR local printer
crc1		CRC1 local printer
dana		DANA local printer
frze		FRZE local printer
sph2		SPH2 local printer
tsg		TSG local printer
ugls		UGLS local printer
ushp		USHP local printer
mnt.ub1	AF79	jobs to be sent to UB
mnt.rdr1	AA7E	recieves jobs coming to UM
umichub	SA17	connection to UB
ub$out	SA17	sends jobs to UB
ub$in	SA17	receives jobs from UB
umichrly	SA19	connection to UMICHRLY
rly$out	SA19	sends bitnet jobs to UMICHRLY
rly$in	SA19	receives bitnet jobs from UMICHRLY
merit	SA18	connection to MERIT bitnet site
merit$ou	SA18	sends bitnet jobs to MERIT
merit$in	SA18	receives bitnet jobs from MERIT
umiphys	SA04	connection to UMIPHYS bitnet site
phys$ou	SA04	sends bitnet jobs to UMIPHYS
phys$in	SA04	receives bitnet jobs from UMIPHYS
sa05	SA05	rje
sa06	SA06	rje
sa08	SA08	rje
sa09	SA09	rje
sa10	SA10	rje
sa11	SA11	rje
sa12	SA12	rje
sa13	SA13	rje
sa14	SA14	rje
sa15	SA15	rje
sa16	SA16	rje
rm1.pr1	SA06	DBRN, tn printer
rm1.pr2	SA06	DBRN, tn printer
rm1.rd1	SA06	DBRN, reader
rm1.rd2	SA06	DBRN, reader
rm5.pr1	SA05	DBRN, tn printer
rm5.pr2	SA05	DBRN, tn printer
rm5.rd1	SA05	DBRN, reader
rm5.rd2	SA05	DBRN, reader

List of RM tasks

The RM uses a number of UMMPS tasks to do its work. Usually, each processor has one UMMPS task. There are also a few central tasks that are always present if the RM is running.

There is one task for communicating with the OPERATOR console, with the name “RM”. The other RM tasks have names of the form “RM.xxxx”. There is also an MTS task for each executing batch job.

The tasks currently used by the RM are as follows:

Task name	**# with this name**	**Devices owned**	**Function**
RM	1	None	operator communication
RM.MAIN	1	None	central task for the RM
RM.LOGR	1	None	writes RM log file
RM.CKPT	1	None	writes RM checkpoint file, which keeps track of moment-to-moment activity, so that the Resource Manager can pick up where it left off (“warm start”) after a stoppage or system crash.
RM.PTR	1 for each started printer	A printer	runs the printer
RM.RDR	1 for each started reader	A reader	runs the card reader
RM.PCH	1 for each started punch	A punch	runs the punch
RM.RJE	1 for each SDA line	SDA line	runs the SDA line
RM.LINK	1 for each SDA line	SDA line	runs the bitnet line
RM.ACC	1	None	accounting reconciliation after printing

List of RM attributes

The job attributes that are currently defined are listed below. Use the SYMBOL command to check the validity of entries in this list. If you discover an error, please see that it gets fixed!

Attribute	Values	Description
delivery	argus1	
	argus2	
	argus3	
	argus4	
	bsad	
	ccstaff	
	ccwest	
	counsel	
	cntr	
	cpha	
	crc	
	dbrn	
	dsc	
	dumwait2	
	dumwait3	
	epa	
	flnt	
	inside	
	isr	
	isr1 - isr7	
	libmvs	
	mail	
	mdph	
	merit	
	mpro	
	mr_list	
	mvsxa	
	n.i.b.c.	
	nubs	
	operator	
	ricks	
	sph0 - sph7	
	tri	
	unyn	
	va.hsrd	
	vdc	
	vri	
devicetype	line	line printer (3211, Memorex 4303, . . .)
	page	Xerox 9790 or 4050 or 9700
	2700	Xerox 2700
	4045	Xerox 4045
	4050	Xerox 4050

	9790	Xerox 9790
	hplj	HP LaserJet series II
	9700	Xerox 9700
	ptrx	printronix
	raw	network attached printer
	qms	QMS Lasergrafix
host	um	execution host 3090 side 0
	ub	execution host 3090 side 1
	hg	execution host Human Genetics
	mt	MTSXA vm test system
overlay	none	
	default	default is none
	lines	or lined
	shading	or shaded
paper	anystandard	plain or 3-hole
	plain	
	3hole	
	label24	24-up label stock
	label33	33-up label stock
	sp1 - sp12	user provided papers
	libcirc	UM Library circulation stock
	librecall	UM Library book recall stock
print	pn	pn print train (line printers)
	tn	tn print train (line printers)
	tn37	tn post t-day print train (line printers)
	ala	ala print train (line printers)
priority	normal	
	low	
	deferred	
	minimum	
proute	argi	Xerox 4050, Computing Center at 535 W. William
	cntr	North Campus Computing Center
	nubs	Xerox 9790, North University Building
	unyn	Xerox 9790, Michigan Union
	arg1	535 W. William, first floor
	arg2	HP Laserjet, 535 W. William, second floor
	arg4	535 W. William, fourth floor
	arox	Xerox 4045
	chur	Xerox 4045
	crc1	Xerox 4045
	dana	Xerox 4045
	frze	Xerox 4045
	sph2	Xerox 4045
	ugls	Xerox 4045
	ushp	HP Laserjet
	dbrn	line printer, Dearborn
	flnt	line printer, Flint
	soph	line printer, School of Public Health
	fadm	line printer, Flint Admin.
	mdph	line printer, Mi. Dept. of Public Health
	ford	line printer, Ford world headquarters
	mlrc	line printer, Medical School Learning Resource Center
	mvel	line printer, EPA

	meds	line printer, Medical School Information Systems
	test	line printer, testing
rerun	yes	
	no	
sepcopy?		
shift	yes	
	no	
special?		
twosided	yes	
	no	
bytes	numeric	
cards	numeric	
cart_drives	numeric	number of cartridge tape drives
copies	numeric	
delay	numeric	processor parameter
images	numeric	
job#	numeric	
pages	numeric	number of images
part	numeric	for large, multiple-part print job
pcopies	numeric	
sheets	numeric	
tapedrives	numeric	
time	numeric	
vm_priority	numeric	priority on TAG record
class	string	
address	string	
comment	string	header sheet comment field
dest	string	bitnet destination
file	string	bitnet file name
jobtime	string	
linecount	string	
proj	string	project number
domain	string	
initial_vars	string	
mailbox	string	
margin	string	for 9700-type printers
proc	string	specific processor name
print_info	string	extended print route info
punch_info	string	extented punch route info
remote_device	string	
sender	string	
sitename	string	
source	string	device name or bitnet source address
user	string	ccid
username	string	user's name
vmargins	string	
vm_id	string	VM id to spool job to
vm_name	string	"filename filetype" for CP CLOSE
vm_node	string	VM node to send job to
vm_site	string	VM site to send job to

Priorities

Each execution job is given a priority based on its time estimate and rate class. The priority of each print job is determined by the number of images it will print and its rate class. The priority of a print job is raised by 1 for each two hours it waits, up to the maximum priority for that rate class. The priorities are 0 through 15.

Jobs are selected in decreasing order of priority. Within a priority, jobs are selected FIFO. Note that a print job can be selected only if some print processor with the right set of attributes is active (e.g., a print job requiring LIBCIRC paper will be selected only if some printer currently has that paper), so jobs do not always print strictly "in order".

The table below shows the execution and print priorities for jobs at normal rates. The priority of a job at low rates is 2 less than it would have at normal rates, at deferred rates 3 less, and minimum rates 4 less.

Priority	**Execution time (second)**	**images to print**
15		≤5
14		≤10
13		≤20
12		≤50
11		≤100
10	≤1	≤200
9	≤2	≤350
8	≤4	≤500
7	≤8	≤650
6	≤16	≤800
5	≤32	≤1000
4	≤64	≤1500
3	≤128	≤2000
2	≤256	≤4000
1	>256	≤6000
0		>6000

The priority table may be changed from time to time. If you find that this table is out-of-date, please see that it gets fixed.

Control and sense commands

Below is a list of control commands accepted by various processors. This is all we know about this subject. If you know more, please see that your knowledge is used to improve this section.

The following table lists the known control commands.

Processor	Command	Notes
RJE	CONTROL=	
	CTL=	sent to line DSP
reader	CANCEL	
printer	REDO	
	BACKSPACE	
	INTERRUPT	
	CANCEL	
	FORWARDSPACE	
	SINGLESPACE	
purge	DRAIN	
	REDO	
	DELAY=n	
punch	CANCEL	
processor		
	REDO	
message	EXPORT=	ON or OFF, passed to msgling
	IMPORT=	ON or OFF
	MAX_SIZE=n	
link	CONTROL=	
	CTL=	
	RNM=	
generic output		passed to the “ling”
execution	CANCEL	
	REDO	
delete	DRAIN	
	DELAY=n	
sysout	CANCEL	
	REDO	
	DRAIN	

The following table lists the known sense commands.

Processor	Command	Notes
all	TASK	returns processor's UMMPS task number
RJE	REMOTE_TYPE	
	REMOVE	remote name
	TYPE	remote type
	SENSE=	or SNS=, passed to line DSP
reader	none	

printer	none	
purge	DELAY	
punch	none	
message	MSGSTATUS	passed to the “ling”
link		pased to DSP
generic output	passed to the “ling”	

Problem handling

How to tell if something is wrong

1. Use SYSTEMSTATUS to enquire about the RM tasks
   $SY Tasks RM (or $SY T R)
   should print something like (on UM):

```
00032  RM.LOGR 00076AA0   19 Monitor Idle
00033  RM.MAIN 00076D60  361 Monitor Idle
00034   RM.ACC 00076EC0   17 Monitor Idle
00035  RM.CKPT 00077020  128 Monitor Idle
05318  RM.LINK 000AD520   91 Monitor Idle; SA17
02297  RM.LINK 000ADD60   92 Monitor Idle; SA07
```

   There are one each of the logger, main, accounting, and checkpoint tasks. The LINK tasks are for the BITNET links, and there is one for each link. Currently, SA07 is the WAYNEST1 link to Wayne State, which is our connection to the rest of BITNET. SA17 is UMICHUB which is the link to the UB machine. On UB, the same first four tasks will appear, and there should be one LINK task, with SA00 attached. This is UMICHUM and is the other end of the UM to UB BITNET link.

   Currently, the links and their SDA lines are:

```
on UM: WAYNEST1  SA07  link to Wayne State VM machine
                       our link to the rest of BITNET
       UMICHUB   SA17  link to UB machine
       MERIT     SA18  link to NSFNET VM machine
       UMDSCVM   SA19  will be link to DSC
                       not yet operational
on UB: UMICHUM   SA00  link to UM machine
```

   To find out which SA line is attached to which link, look at the file SFIL:DEFINITIONS. This file is read by the RM at startup time and used as commands to modify its symbol table (the source for the symbol table itself is in RMGR:SYMTABLE>SD, but I would not advise studying this if you value your sanity.). Later definitions and redefinitions can be done at any time by feeding these commands to the SYS:RMOPR program. Each link has three processors defined for it, as for example

```
define device=sa07 waynest1              type=link   link=waynest1
define device=sa07 wsu$out               type=sysout link=waynest1
define device=sa07 wsu$in par=(stream=any) type=sysin  link=waynest1
```

   You can also find out this information by displaying the

definitions using the SYS:RMOPR program. See next section.

Outgoing BITNET traffic will be found in queues the same name as the link: WAYNEST1, UMICHUB, ...
Incoming BITNET traffic ends up in the queue MESSAGE if it was mail or in the queue IMPORT if it was a file.

The normal (good) case is for all the outgoing queues to have no entries, and for the MESSAGE queue to also have no entries. The IMPORT queue will normally be quite large (over 100) as the jobs that represent incoming file transfers stay there until the user copies *IMPORT* into their own file. Most other queues are currently zero or very small, except for NETPURGE and MSGPURGE, which are usually very large because they are defined with a 24-hour retention period, and DELETE, which has an 8 hour retention period.

In this case, you should check again after a while to make sure the job number the output processor is handling changes. Otherwise, the link is down or otherwise incapacitated. Note that the main processor for a link will always be waiting for work, since it doesnt do anything but direct the other processors.

Fixing a stuck processor

The basic idea is to stop the processor (or processors) that is not running, and start a new one. If you need to stop a processor that you cannot “reach” yourself, (because, say, it is at Wayne State), contact the NOC and have them contact the appropriate people.

For clarity, these instructions will use an example. Suppose that the “stuck” processor is UMICHRLY, running line SA07.

First, DRAIN UMICHRLY. The RM should respond:

```
Processor UMICHRLY drained.
Processor UMICHRLY destroyed.
```

Continue even if you do not get this response. Next, use systemstatus to check that the task is really gone: /t d SA07. If the response looks like

```
00634  RM.LINK 000A0D00   91 Monitor Idle; SA07
```

KILL or STOP (via systemstatus or the operator's console) the job. The job may take a few minutes to disappear; wait a bit before resorting to more drastic measures.

Finally, issue an RM command to start a new processor: $START UMICHRLY. If the new task immediately gets a program interrupt (and requests a dump), it is likely that the old task was not really gone. Cancel the dump and try again, making sure that the old task is gone.

Restarting the whole RM

4. It sometimes happens that things get really hosed up, and you need to get rid of the whole RM and restart it. For this case, use the predefined MTS files to drain and restart it so if we add new steps to the startup we don't have to find every last copy of this note and modify it. What you should do is:

   a. Use operators' console (or $SY O ...) and issue
      MTS *RMD
   ("D" for Drain)

   b. It should print a message on the operators'console that the RM was drained. If nothing happens, do a /T on the MTS *RMD job and see what it is doing. If it is sitting in "Monitor wait" status, the RM intertask network is probably also hosed up, so proceed to (5) below.

   c. Use the systemstatus command TASKS RM (or /T T RM) to make sure all tasks have gone. (Be patient)

   d. Make sure the messagesystem task connected with all of this has gone. Either look for a job that has POST:NETLOG.BIT locked (should be id POST and *RM.M showing as front of msource name when SY displays it (actual name is *RM.MSG)) or else use SYS:TASKS and enter INTERTASK RMGRNET to find it. If you find it, stop it.

   e. Issue
      MTS *RMS
   to start up the system

   f. and then issue
      MTS *RMP
   to start up the processors

5. If you try running SYS:RMOPR and it immediately produces three lines of error messages from someone (the subtasking monitor?) and then sits in "Monitor wait" limbo, the communications path to the RM is also hosed up. (The "Monitor wait" condition is the system equivalent of the "have a happy day" face with "What me worry?" printed under it when you're dealing with programs that use the subtasking monitor)

   a. Run SYS:TASKS
   enter the command KILL RM

   b. then (still under SYS:TASKS)
   enter the command TASKS RM
   to make sure they're all gone. Wait till they are.

c. then (still under SYS:TASKS)
enter the command INTERTASK RMGRNET
If it finds any tasks, KILL them.
(There will probably be one task, the *RM.MSG task)

d. Now things should be cleaned up and you can restart the RM in the normal way with the operator files:
MTS *RMS
MTS *RMP

e. Even though the RM has started again, a given link may not be running again (because both ends need to be drained and then both ends started). So check to see if any of the links have nonzero queues, and if so watch to see if the counts go down.

6. If the processor BITMSG keeps reporting that it is "still initializing" then there is a leftover copy with POST:NETLOG.BIT locked still around. Use the systemstatus command to find all MTS jobs with the id POST:
SY T U POST
You should find two tasks showing an msource name of "*RM.M". The newer task (one with the larger task number) should report FILE WAIT. If this is the case, stop the older task (the one with the smaller task number). This will allow the other to finish initialization.

Disappearing/dying processors

If one processor (any type) repeatedly vanishes or causes program interrupts, it is probably unable to process a particular job. The basic strategy for this problem is to find the offending job and put it in hold.

If the processor gets a program interrupt, the problem job is probably the one it was processing at the time. Use the RM PROC command to get the job name and number. Put the job in HOLD. STOP or KILL the processor (using systemstatus or the operator's console). Make sure the processor is gone (but wait a few minutes before taking drastic measures). Use the RM START command to start the processor back up.

*** if get repeated interrupts *** ***

If a processor disappears, issue a new RM START command for it. If it keeps disappearing, the problem job usually is requeued at the end of the queue when the processor dies. However, by the time you look at the queue, newly-arriving jobs may have been added after that job. Use the RM QUEUE command to list the jobs in the queue. If you see a good candidate for the problem job (e.g., there is only one job), put that job in HOLD. Issue a new (RM) START command for the processor. Make note of the contents of the queue. If the processor continues normally, you have the culprit in hold. If the processor stops again, look at the

queue. The problem job is one which was also in the queue before. It is probably the job that was closest to the front of the queue the first time you checked. Put that job in HOLD and START the processor. Continue in this fashion until you do identify the problem job. RELEASE any other jobs you have put in hold.

Once you have identified a job that is causing interrupts or disappearances, leave it in hold. Contact the firefighter or RM-caretaker, and ask that person to look at the job.

Problem jobs

This section explains how to determine whether a job has a bad spoolfile. It is intended for firefighters.

When inspecting spoolfiles, remember that you are reading someone's mail or data. Read as little as possible, and destroy (or give to the owner) any printed copies you may create.

First, get the name of the spoolfile(s). Use the RM SPOOLFILES command (use SYS:RMOPR). The files will have names of the form SPOOLFILxxxx. The files are on id SFIL, and not permitted.

The most common problem can be checked with the MTS editor. See if the first line is 16 bytes long. If it is, the file is bad.

If the file passes that check,

```
RUN RMGR:PRTSPFL>OQ SCARDS=SPOOLFILxxxx
SPRINT=-p
```

If you get a program interrupt, the spool file is bad.

*** *** what next

Index