NCR CONTROL DATA CORPORATION
ADVANCED SYSTEMS LABORATORY

CHAPTER 04

JOB MANAGEMENT

Doc. No. ASL00282
Rev. 04

Copy No. 87

Rev 27 May 75

TABLE OF CONTENTS

1.0 INTRODUCTION

The job is the vehicle through which the individual user interfaces to the IPL Operating System. A job is comprised of a set of user defined function/environment definitions and a set of formal system created structures. The definitions provided by the user occur in the form of system command language statements which describe and direct a computing process. The system created structures permit the operating system to uniquely identify, monitor, and account for the user computing session.

Users may present job definitions to the operating system from either batch input devices or from interactive terminals. In the first case a complete sequence of command language statements is transferred from the input device to system mass storage for subsequent processing. In the second case command language statements are interpreted on a statement-at-a-time basis as they are read from the terminal.

Job Management is concerned with the following areas:

Recognizes that a terminal or a batch input device is ready for servicing;

Establishes a basic environment within which the job will operate;

Schedules the use of computing facilities among multiple Jobs;

Distributes job-generated output files to user specified destinations;

Controls the allocation of peripheral resources to active jobs;

Returns system resources from terminated jobs and;

Accounts for the system resources used by jobs.

The normal mode of operation of the IPL Operating System is to have as many jobs active in the system as an installation will permit. All scheduling, resource reservation, and operational

decisions are made by the job through command language statements as it executes. It is assumed that all jobs are active and scheduling decisions are made as a result of command language requests which are described elsewhere in this and the command language documentation. Jobs which are queued due to system saturation are removed from the queue and placed into execution on a first in first out basis.

1.1 DEFINITION OF TERMS

Command Language - the language through which an external user communicates with the system.

Command Language Interpreter - the system provided routine which is responsible for recognizing and interpreting statements written in command language.

Control Language - the SWL language macros through which an executing program communicates with the system.

End of Physical Input Stream - a logical status pertinent to the system input stager. This status occurs only when a system input device signals "end of data" and the final record in the input buffer is a fence record.

Fence Record - a system defined record of a unique configuration which is placed at the end of a logical input stream to identify the end of that logical input stream.

Job Establishment - the process of creating basic and fundamental structures for identifying and controlling a new job. Job Establishment functions are performed by the Job Establisher task in the System Job.

Job Initiation - the process of expanding the local environment of a previously established Job. Job Initiation is performed by the Sequence Monitor task in each individual Job.

Logical Input Stream - the volume of input data which appears between fence records in a physical input stream.

Physical Input Stream - all input data which is available to the system via an active system input device. A physical input stream is comprised of one or more "logical" input streams.

Primal Invocator Identifier - the ultimate terminal or site which caused the execution of a job. The primal invocator identifier (PII) of an interactive job is the terminal. If the interactive job SUBMITs a batch job, the batch job's PII is also the terminal. The PII is used to identify the logical operator of a job and for default file routing information.

Primary Input File - the source from which an instance of the Sequence Monitor obtains command language statements when no alternative source is specified by the command language. The name of the Primary Input File is information required by the Sequence Monitor from the routine which invoked it. A disk file is the primary input file for a batch job. The primary input file for an interactive job is the terminal.

Resource, non-preemptible - a non-preemptible resource is one which cannot be taken away from a job without cooperation from the job. Non-preemptible resources include files, volumes and units.

Resource, preemptible - a preemptible resource is one which can be taken away from a job, used by another job, and returned with no effect on the first job except execution time. Preemptible resources include memory and CPU's.

Sequence Monitor - a system provided program which serves as the primary control element during the interpretation of a logical input stream. The command language interpreter comprises a part of the sequence monitor.

Staging - the process of transferring logical input streams from system input devices to mass storage files. Each logical input stream is placed on a unique mass storage file.

System Input Device - an input device which the system defines as being dedicated to the process of submitting "physical" input streams.

System Input Stager - the system provided routine which is responsible for staging.

1.2 JOB SUBMISSION

1.2.1 FROM EXTERNAL SOURCES

Jobs may be introduced in one of two ways based on the classification of the initiating device. If the device is classified as a system input device then its function is the submission of batch jobs. If it is classified as an interactive terminal then its function is the submission and control of interactive jobs. A device's mode of operation is determined by installation parameters and the functional attributes and capabilities of the device.

1.2.1.1 System Input Devices

When a device classed by an installation as a system input device becomes active the input stager is notified. This routine will create a temporary mass storage file as a destination for the job deck being entered from the device. When an end-of-file is detected the input stager makes the temporary mass storage file containing the job deck a permanent mass storage file. This permits the system to remember files that have been completely staged across a system crash. The input stager next prepares to submit the job to job management.

If the submit request is rejected due to any type of system saturation, the job is queued for subsequent execution.

1.2.1.2 Interactive Devices

When an interactive terminal becomes active either through a dial-in or initiation of activity on an inactive terminal, the System Access Manager is notified. The System Access Manager prepares to submit a job with the terminal as the primary input file.

If the submit is rejected due to system activity, the user will be notified at his terminal to try again at a later time. The terminal will then be closed and disconnected if on a dial-up line.

1.2.2 FROM ACTIVE JOBS

The capability is provided by which active jobs may invoke the asynchronous processing of other jobs. To utilize this capability the active job must provide a file of command language statements on a permanent mass storage file and then request the creation of a job which will utilize the file as its primary input file. Jobs created in this manner are initiated as batch jobs.

1.3 JOB INITIATION

The process of job initiation occurs within an established job. The process commences immediately after the activated instance of the Sequence Monitor program gains initial control and continues until such time that the Sequence Monitor determines that a non...LOGIN statement may be accepted from the primary input file associated with the job. Although control of the job initiation process resides with the Sequence Monitor program and the functions performed by the process are performed independently of the individual user, installation and user participation in the process is not precluded. The capability for such participation is provided by user and installation oriented profiles (refer to section 1.7).

1.4 JOB TERMINATION

Job Termination is a system-supplied facility which provides the functions and the control required for the systematic removal of previously initiated jobs from the system. The job termination process executes within the individual job under control of the cognizant instance of the Sequence Monitor program. The job termination process is invoked within a job as a result of one of the following situations:

1. A command language logout statement is encountered by the command language interpreter in the command sequence which it is currently servicing, or an end-of-file condition is encountered by the command language interpreter while processing statements from the primary input file.

2. A command language login statement is encountered subsequent to job initiation (i.e., a second login command) by the

command language interpreter in the command sequence currently being serviced.

NOTE The logout capability is also provided via control language. The second login capability is not.

1.5 OUTPUT FILE DISTRIBUTION

Output file distribution is a system-supplied facility which transfers user-specified output files to user-specified destinations (printers, punches, terminals).

The routing mechanism provided by Job Management is designed to facilitate the transfer of data between any file in the system and any output device. Obvious incompatibilities will occur such as attempts to output libraries of object code. It is intended that the routing facilities provide standard conversion utilities to reformat the data in such a way as to provide useful output.

The three Task Services requests which provide the user interface are JM#ROUTE, JM#DIRECT, and JM#RETRACT. There will also be a number of operator control mechanisms which permit the operator to achieve output unit setup and control when such action is needed. The latter set of directives are outside of the user repertoire and will be available only to programs which act at an operator's level of authority.

The form control parameter, which is a string of up to 32 characters, is used to specify the physical properties of the output medium (eg. paper size, card type, printer train, ribbon color) upon which the data is to be placed. The actual meaning of the form parameter and the resulting interaction between the user and the operator in control of the setup of the output units will be left to the discretion of each individual site. The operating system will be designed to support the operator in that it will remember the current form setups of all output units and will facilitate a change from one setup to another. It is intended that algorithms will be developed which will notify the operator that a form change is required based on the priority of the files in the output queues and the number of output units available which can accept the required form.

One problem which is not dealt with at the level of Task Services is the question of a system crash. The loss of the output queues should not be allowed to negate the transfer of these files. However, the use of a file to retain the information in its original form of output queues and File Router tables may not be practical. Other methods should be investigated which will preserve the information in the event of a system failure. It is recommended that a procedure be adopted which posesses as secure a recovery mechanism as possible and which requires the minimum of system overhead to save the information necessary to restore the output queues (assuming the information will never or very infrequently be used). One possible method would be to log each route request using a unique identifier. When the transfer was complete, the accounting information would also include the unique identifier. A missing echo would be used to signal the recovery programs that the file had not been transferred.

1.6 JOB FLOW

1.6.1 OVERVIEW OF JOB MANAGEMENT IN SYSTEM JOB

Following is a list of names of programs which execute under the auspices of the System Job and provide functions which are relevant to job management. For each program listed, there is a definition of its structure and an overview of the functions which it performs.

A graphic overview of the flow of data and control among these listed job management programs is provided via the attached diagram entitled "Job Flow Through the System Job" (Figure 1.6.1-1). Circled letters on the diagram correspond to the underlined letters which appear in the functional overviews of the programs; these letters are provided as points of reference.

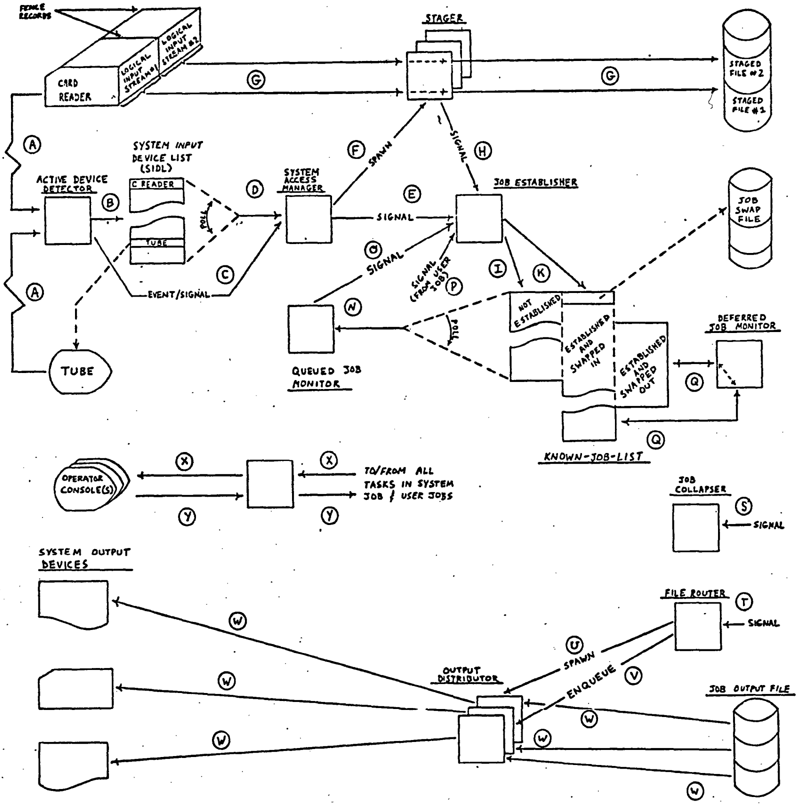
Programs of System Job

Name	Structure	Functional Overview
Active Device Detector	a part of Configuration Manager	- Detects hardware level signals A which indicate that a previously inactive device has become active. - Associates a particular hardware signal with a particular device and places the device type and device identifier in the System Input Device List B - Causes an event or sends a signal C to awaken System Access Manager.
System Access Manager	Task	- Polls System Input Device List for an entry which identifies a device requiring service D - When no entries exist, waits for event/signal C - When entry exists in System Input Device List, makes a determination of whether the device is for batch or interactive. - For interactive, issues JM#SUBMIT which results in a signal being sent to Job Establisher E - For batch, invokes the execution of a Stager subtask F (i.e., issues a PM#SPAWN request)
Stager	Subtasks of System Access Mgr.; one active per every active batch System Input Device	- Copies a Logical Input Stream from a batch System Input Device to a permanent disk file G - Issues JM#SUBMIT which results in a signal being sent to Job Establisher H - Repeats above for every Logical Input Stream on the batch System Input Device, then terminates execution.
Job Establisher	Task	- Awakened by arrival of a signal E, H,O, or P which contains information concerning a job for which establishment is desired. (Signal P results from the issuance of a JM#SUBMIT request from within a user job.) - If system limits prohibit immediate job establishment, and if the job is interactive (or is batched with "no enqueue" specified) sends a reject signal to the requestor. - If system limits prohibit immediate job establishment, but the job is batch with enqueue permission specified, creates a Known-Job-List (KJL) entry and marks it as "not established" I - When immediate establishment is permissible: Uses the System Monitor provided SM#CREATE_ADDRESS_SPACE request which constructs fundamental Job tables and segments. Acquires a Known Job List entry, sets the entry's status field to "Established-swapped in", and interlinks the KJL entry and the Job Control Block (JCB) for the new Job. Triggers the execution of the Sequence Monitor Program in the new Job.
Queued Job Monitor	Task	- Awakened by a time event - Polls Known-Job-List entries marked as "not established" for the one having the highest priority N, then sends a signal O, to Job Establisher; the signal includes the identity of the JCB associated with the Job to be established.
Deferred Job Monitor	Task	- Awakened by a time event - Performs swap-out/swap-in of jobs based upon scheduling algorithms and system state;

		swap out includes deallocation of system segments assigned to a job and writing of a job image to a swap file; swap in includes allocation of system segments and reading of the job image from the swap file into memory. Known-Job-List entries are appropriately marked Q to indicate whether a job is swapped-in/swapped-out.
Job Collapser	Task	- Awakened by signals which originate from within the job termination procedures of user jobs S - Uses information accompanying a received signal to identify a specific job which is to be collapsed - Deallocates segments, tables, etc., which are assigned to the job, and which cannot be deallocated by the job termination procedures in the user job
File Router	Task	- Awakened by signals I from the JM#ROUTE request processors in the System Job and in user jobs - Determines identity of file to be routed and the desired routing destination from information accompanying the signal - Determines whether output to the desired destination is currently active; if it is not, invokes the execution of an Output Distributor subtask, via PM#SPAWN request, and specifies the identity of the file to be routed U; if output to the desired destination is already in progress, places name of file to be routed in a queue for the relevant subtask of the Output Distributor V

Output Distributor	Subtasks of File Router; one active per each active System Output Device	- Opens a System Output Device - Transfers a file from disk storage to the System Output Device W - Closes the output device when no additional files are queued for transmission to the System Output Device currently being serviced; terminates execution
Operator Communication	To be defined	- Receives messages from tasks in the system job and from tasks in the user jobs, and transmits those messages to the operator X - Receives messages from the operator and transmits those messages to tasks in the system job or to tasks in user jobs Y

1.0 INTRODUCTION
1.6.1 OVERVIEW OF JOB MANAGEMENT IN SYSTEM JOB



Job Flow Through The System Job

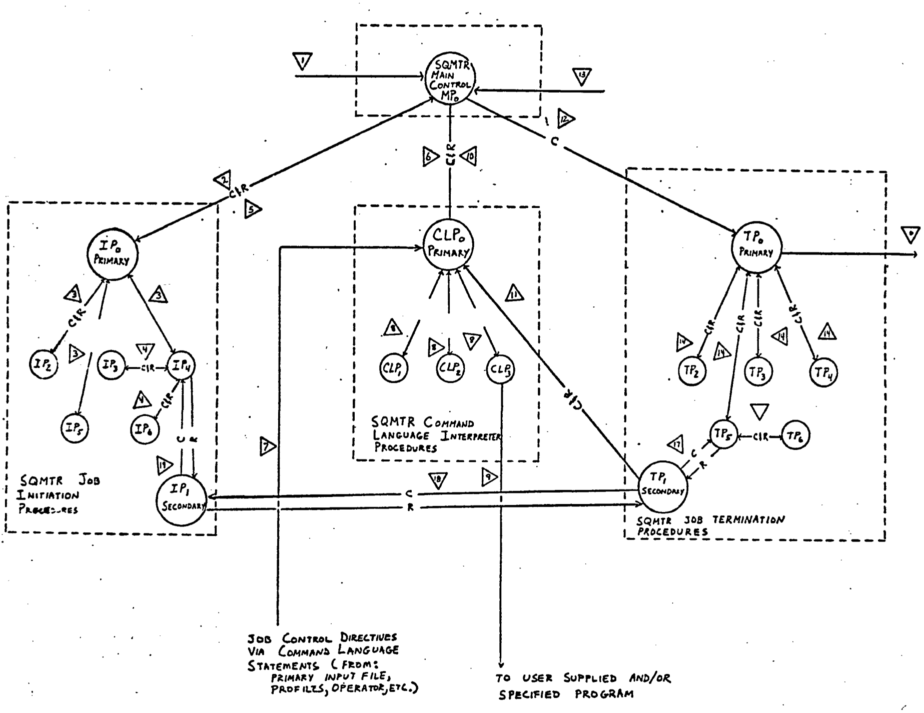
Figure 1.6.1-1

1.6.2 OVERVIEW OF JOB MANAGEMENT IN USER JOB

The controlling element of every user job is a system provided program - SEQUENCE MONITOR. This program, which executes as a task in the address space of every user job, is constituted of numerous procedures; these procedures are logically categorized into four sets of procedures: Sequence Monitor Main Control, Job Initiation, Command Language Interpreter, and Job Termination. A general discussion of the responsibilities/functions of each set of procedures within the Sequence Monitor program follows.

<u>Procedure Set</u>	<u>Responsibilities/Functions</u>
Sequence Monitor Main Control	o Provides the point of initial job execution following job establishment o Provides sequencing control over the other three sets of procedures in Sequence Monitor o Provides "interrupt" location and logic to permit atypical job termination (e.g., forced termination by operator or operating system)
Job Initiation	o Expands the basic environment of the job as initially provided by the job establishment process; e.g., creates standard job output files, connects files to streams
Command Language Interpreter	o Validates Jser and account identifiers o Performs LOGIN processing o Processes command language statements and directs job processes as specified by these statements
Job Termination	o Reduces the job environment to a level which is equal to, or slightly greater than, the level provided initially by job establishment; e.g., closes job files, routes standard job files, releases job local LNS segment o Advises Job Collapser in the System Job when conditions dictate that the user job is to be completely purged from the system

The attached diagram entitled "Overview of Sequence Monitor" (Figure 1.6.2-1) is provided as a graphic representation of the types of inter-set and inter-procedure flow of control which occurs within the Sequence Monitor program. Large and small circles on the diagram represent non-specific procedures, i.e., they are presented to depict <u>types</u> of interrelationships only, and are not intended to imply any formal procedure name - procedure function relationships. The letters C and R contained within the lines representing control flow indicate procedure calls and procedure returns respectively. The discussion which follows is an explanation of the program flow as depicted by the aforementioned diagram.


SQMTR MAIN CONTROL MP_0
1
13
12
C
6
C/R
10
2
C/R
5
IP_0 PRIMARY
3
C/R
3
IP_2
3
IP_3
4
C/R
IP_4
IP_5
4
C/R
IP_6
C
R
19
IP_1 SECONDARY
SQMTR JOB INITIATION PROCEDURES
CLP_0 PRIMARY
8
8
9
11
CLP_1
CLP_2
CLP_3
SQMTR COMMAND LANGUAGE INTERPRETER PROCEDURES
7
18
9
C
R
C/R
TP_0 PRIMARY
14
C/R
C/R
C/R
C/R
14
14
14
14
TP_2
TP_3
TP_4
TP_5
C/R
TP_6
17
C
R
TP_1 SECONDARY
SQMTR JOB TERMINATION PROCEDURES
JOB CONTROL DIRECTIVES VIA COMMAND LANGUAGE STATEMENTS (FROM: PRIMARY INPUT FILE, PROFILES, OPERATOR, ETC.)
TO USER SUPPLIED AND/OR SPECIFIED PROGRAM

Overview of Sequence Monitor

Figure 1.6.2-1

Diagram Reference Numbers	Discussion
1	Sequence Monitor Main Control gains control the first time following job establishment; this is accomplished by presetting various elements in the Control Point used to control the execution of Sequence Monitor.
2	Sequence Monitor Main Control (MP(o)) calls the primary procedure (IP(o)) of the set of Job Initiation procedures.
3, 4	Procedure IP(o) issues calls to invoke the execution of all subsidiary procedures which results in *all* functions of job initiation being performed.
5	IP(o) returns to MP(o)
6	MP(o) calls the primary procedure (CLP(o)) of the set of Command Language Interpreter procedures.
7,8,9	Command Language Interpreter performs LOGIN processing, user and account identifier validation, and command statement processing by reading command statements 7 and calling subsidiary procedures 8 to perform required functions. Functions performed may include invoking the execution of a user provided and/or specified program 9
10	Upon encountering a command language LOGOUT statement CLP(o) returns to MP(o) 10
11	If, after having completed LOGIN processing, Command

Language Interpreter encounters a subsequent LOGIN statement in the command stream, CLP(o) calls 11 a secondary Job Termination procedure (TP(1)) to effect partial job termination and job reinitiation. Upon return of TP(1) to CLP(o), Command Language Interpreter continues as described for 7, 8, and 9

12,13 MP(o) calls the primary Job Termination procedure (TP(o)) to effect total job termination with no subsequent job reinitiation. This call, for which no return ever occurs, occurs as a result of CLP(o)'s return 10 after MP(o)'s call to it 6, or an "interrupt" 13 into Sequence Monitor Main Control for atypical activation of Job termination.

14, 15, 16 Procedure TP(o) issues calls to all subsidiary procedures, which results in all functions of Job termination being performed. Upon return of all subsidiary procedures, TP(o) sends a signal 16 to Job Collapser in the System Job; following completion of the services of Job Collapser in response to the transmitted signal, this job will no longer exist in the system.

17, 15, 18 The secondary Job Termination procedure (TP(1)), in response to the call 11 from CLP(o), issues calls to selected subsidiary procedures 17 and 15 to effect only a partial job termination process. Having effected partial job termination, TP(1) issues a

call 18 to the secondary Job Initiation procedure (IP(1)) to re-expand the job's environment to its usual level following job initiation. Partial initiation is required because only partial termination was performed. Upon return of IP(1), TP(1) returns to CLP(o).

19, 4 The secondary Job Initiation procedure (IP(1)), in response to the call 18 from TP(1), issues calls to selected subsidiary procedures 19 and 4 to effect re-expansion of the job's environment to its usual level following job initiation.

1.7 PROFILES

A major facet in the tuning of the operating environment for a particular user is profile processing. Profiles are files of command language statements which are accessed via a system profile directory. They are implicitly executed by the Command Language Interpreter as various levels of user identification are achieved. Their interpretation is analogous to that caused by the ENTER command language statement.

When the Command Language Interpreter is first entered, it executes a System Profile, if any. This can set installation default conditions and limits on all system users.

Next, the profile appropriate to the terminal (or system input device) and port are processed.

After this, the LOGIN is processed to determine the account and user identifiers. Profiles corresponding to these identities are processed in that order.

Finally, interpretation commences on the primary input file for the job.

All functions and system services which are available through normal command language sequences are also available through profile processing: e.g., through the use of profiles, standard environmental requirements such as priorities, various limits, mounting of removable volumes, attaching/detaching/opening/closing files, dialogue with the operator, etc., can be fulfilled implicitly as part of the login process.

1.8 SAMPLE JOBS

```
JOB WHJ "COBOL COMPILE AND EXECUTE"
COLLECT SOURCE
.
 .  COBOL SOURCE DECK APPEARS HERE
  .
**
COBOL I=SOURCE, O=OBJECT, L=LISTING, S=ERR
IF ERR.LEVEL GT 0
  PRINT LISTING
  JOBEND
  IFEND
SAVE OBJECT
OBJECT ADD=OBJECT
COLLECT DATA UNTIL = /.
.
 .  DATA DECK APPEARS HERE
  .
/.
EXECUTE PROG=MAIN, PARAM=DATA
JOBEND

JOB WHJ "COBOL EXECUTE ONLY"
OBJECT ADD=OBJECT
COLLECT DATA UNTIL = /.
.
 .  DATA DECK APPEARS HERE
  .
/.
EXECUTE PROG=MAIN, PARAM=DATA
JOBEND
```

2.0 ACCOUNTING

2.1 OBJECTIVES OF IPL ACCOUNTING

- o Detect, measure, and record utilization of system resources by users of the system.
- o Provide consistent information regarding resource utilization for every identical running of a given job.

2.2 DETECTION, MEASUREMENT, RECORDING OF RESOURCE UTILIZATION

2.2.1 DETECTION

Detection of an instance of resource utilization is the responsibility of the system procedure which is chartered to manage the resource; for example, File Manager is responsible for the detection of file creations, releases, etc. while Buffer Manager may be responsible for the detection of file activity.

2.2.2 MEASUREMENT

Measurement of resource utilization consists of providing detail information relevant to the use of each resource; some information may be common to all resources while other information may be peculiar to only one resource. (The discussions of the text portions of accounting records are included in the section on System Logging.) Measurement, also, is the responsibility of the system procedure which manages a resource.

2.2.3 RECORDING

Information regarding each use of a system resource is recorded on a SYSTEM LOG FILE by the appropriate manager of a resource at the time the usage is quantifiable. Recording of accounting information on a SYSTEM LOG FILE is accomplished through use of the JM#SYSLOG request provided by Job Management.

3.0 SYSTEM LOGGING

Logging of system accounting, system error, and system dayfile information is accomplished by system elements through use of the JM#SYSLOG request. The request processor which services the JM#SYSLOG request appends requestor-specified log information to a standard record header and transmits that set of information to a system-provided System Log File.

3.1 LOG FILE CHARACTERISTICS

Any system-provided file which is to be utilized as the System Log File must possess the following characteristics:

- o the file must reside on disk storage;
- o the file must be permanently cataloged and the system defined to be its owner;
- o the file must be sequentially organized;
- o the file's block size must be of fixed length and equal to 4096 bytes; and
- o the file must be dynamically expandable by block-size increments.

3.2 LOG FILE RECORD CHARACTERISTICS

System Log File records are of variable lengths. Each record is comprised of a standard, fixed length Log File record header and a variable length record text portion which is peculiar to the type of record.

3.2.1 RECORD HEADER

The standard Log File record header contains the information listed below.

o Record Identifier Code

A hexadecimal number of the form ttccss, where tt identifies a record type, cc identifies a class of record within a type, and ss identifies a record subclass within a class.

The following values for tt are currently defined.

tt=01	System Bench Mark Record
tt=02	Job Bench Mark Record
tt=03	Accounting Record
tt=04	Error Record
tt=05	Dayfile Record

Specific values of cc and ss are defined in individual discussions of the various record types.

o Current Time

o Length of the record text portion.

3.2.2 RECORD TEXT

3.2.2.1 System Bench Mark Records (Identifier Code=01ccss)

- o Current Date
- o Central Site Identifier
- o Software version/maintenance level

3.2.2.2 Job Bench Mark Records (Identifier Code=02ccss)

3.2.2.2.1 JOB BENCH MARK I

- o Internal Job Identifier (a concatenation of the KJL entry ordinal and the KJL entry sequence number)
- o External Job Identifier (system assigned JCB name)
- o Account Identifier
- o User Identifier

3.2.2.2.2 JOB BENCH MARK II

- o Internal Job Identifier

3.2.2.3 Accounting Records (Identifier code=03ccss)

3.2.2.3.1 VOLUME SET RESERVATION RECORD

- o Internal Job Identifier
- o Volume set name

3.2.2.3.2 VOLUME SET RELEASE RECORD

- o Internal Job Identifier
- o Volume set name

3.2.2.3.3 UNIT SET RESERVATION RECORD

- o Internal Job Identifier
- o Peripheral type
- o Number of units

3.2.2.3.4 UNIT SET RELEASE RECORD

- o Internal Job Identifier
- o Peripheral type
- o Number of units

3.2.2.3.5 FILE CREATION RECORD

- o Internal Job Identifier
- o File name
- o Volume set name on which the file resides
- o Type of peripheral to which the file is assigned
- o Number of bytes of storage allocated to the file

3.2.2.3.6 FILE EXPANSION RECORD

- o Internal Job Identifier
- o File name
- o Account name, User name; if the named file is temporary, these names will be identical to those in the Job Bench Mark I record associated with the Job requesting the file EXPAND operation; if the file is cataloged as permanent, these names will be those registered in the catalog;<u>regardless of whether the file is temporary or permanent, the account name and user name specified here are the ones to which accounting applies</u>
- o Volume set name on which the file resides
- o Type of peripheral to which the file is assigned
- o Previous file size; this is the number of bytes of storage allocated to this file prior to this operation
- o New file size; this is the number of bytes of storage which is allocated to the file subsequent to this operation
- o Date and time of most recent prior adjustment to the size of the file (i.e., creation or expansion); for permanent files this information may be obtainable from the file's catalog; for temporary files this information must be maintained within a file related table and recorded in a catalog if the temporary file is ultimately made permament.

3.2.2.3.7 FILE RELEASE RECORD

The text area of the File Release record is identical to that of the File Expansion record

3.2.2.3.8 FILE ACTIVITY RECORD

- o Internal Job Identifier
- o File name
- o Volume set name
- o Type of peripheral to which the file is assigned
- o Date and time file was opened
- o Date and time file was closed
- o Number of read operations
- o Number of write operations
- o Total number of bytes transferred to and from the file

3.2.2.3.9 JOB TERMINATION RECORD

- o Internal Job Identifier
- o Date and time of job initiation
- o Date and time of job termination
- o Job mode; initial or restart execution (for future consideration)
- o Job class; batch, interactive
- o Termination type; normal, abnormal
- o Number of System Command Language statements processed
- o Primary invocator identifier
- o CPU time used; total number of milliseconds of real time during which control points of the job were dispatched
- o Memory used; an accumulated page-time value

It is recognized that the calculation of memory utilization will not guarantee identical results for every identical instance of execution of the same job; variations will occur because of 1) the effects of sharing upon the calculation of working set size, and, 2) timing variances associated with asynchronous job related activities. Even though providing inconsistent accounting information is in direct conflict with one of the previously defined objectives, such deviation is considered to be justified in this case; memory is a facility of obvious value to the system and charges for the use of it are appropriate. Because of the nature of the IPL system, no practical technique to assure consistency of memory utilization measurement is evident; it is assumed that users will understand the reasons for these possible inconsistencies and accept them.

3.2.2.3.10 FILE ROUTING RECORD

- o Internal Job Identifier
- o File name
- o Date and time routing initiated
- o Date and time routing completed
- o Peripheral type of source device
- o Peripheral type of destination device
- o Destination name
- o Special forms identifer
- o Number of copies routed
- o Number of read operations
- o Number of write operations
- o Total number bytes transferred

3.2.2.4 Error Records (Identifier code=04ccss)

To Be Supplied

3.2.2.5 Dayfile Records (Identifier code=05ccss)

3.2.2.5.1 MESSAGE TO OPERATOR

- Internal Job Identifier (from which message is issued)
- Operator Identifier
- Text of message transmitted to the operator

3.2.2.5.2 MESSAGE FROM OPERATOR

- Internal Job Identifier (for which message is intended)
- Operator Identifier
- Text of message from operator

3.2.2.5.3 FREE-FORM MESSAGES

- Text of any free-form message which any system element desires to be placed on the System Log File; if sender identifier is necessary it is the responsibility of the sender to include such information within the message text.

3.3 ASSIGNED RECORD IDENTIFIER CODES

Following is a list of currently assigned System Log File Record Identifier Codes and the associated record names. This list is tentative because it is recognized that changes of code assignment within a record type may be desirable in order to place a significance on record class and subclass different from those shown; further, it is assumed that requirements for additional records will be discovered as the system design effort progresses. This list, however, is to be considered the focal point for coordinating any definition and/or modification of System Log File Record Identiffer Codes.

Identifier Code			Record Name
TYPE	CLASS	SUBCLASS	
01	01	01	System Bench Mark
02	01	01	Job Bench Mark I
02	01	02	Job Bench Mark II
03	XX	XX	Accounting Records
03	01	01	Volume Set Reservation
03	01	02	Volume Set Release
03	02	01	Unit Set Reservation
03	02	02	Unit Set Release
03	03	01	File Creation
03	03	02	File Expansion
03	03	03	File Release
03	03	04	File Activity
03	04	01	Job Termination
03	05	01	File Routing
04	XX	XX	Error Records
05	XX	XX	Dayfile Records
05	01	01	Message to Operator
05	01	02	Message from Operator
05	02	01	Free-form Message

4.0 SCHEDULING

4.1 DEVICE ALLOCATION AND SCHEDULING

To Be Supplied

4.2 JOB SCHEDULING

The normal mode of operation of the IPL Operating System is for all jobs known to the system to be in some sense active. User validation and command language interpretation are not done until a job has been established and has executed for some period of time. Thus, very little is known about a job before it is executing. Job scheduling is designed to accomodate this mode of operation, where jobs must be preempted as they express requirements for resources and resumed as the resources become available.

Job scheduling is designed to be, as much as is feasible, table driven. Many of the factors which determine scheduling decisions may be set at system generation time and/or varied dynamically while the system is running. This will permit an installation to tune scheduling to its particular mode of operation and to change the mode of operation during the processing day.

Job scheduling must accomodate a mixture of batch, interactive, and transaction jobs simultaneously and in various combinations.

4.3 JOB STATES

A job in the IPL Operating System exists in one of several states which are of interest to the schedulers. These states are: queued, deferred, inactive, and active. There will also be frequent references to the running state which implies inactive or active. These states correspond to the amount known about a job and the amount of system resources used by a job. The system knows the least about a queued job and it uses the least system resources. The system knows the most about an active job and it requires the most system resources.

One of the major functions of job scheduling is to move jobs from one state to another. The following diagram indicates which state transitions are possible and also which scheduler controls that transition.

4.0 SCHEDULING
4.3 JOB STATES

ORIGINAL STATE	FINAL STATE QUEUED	DEFERRED	INACTIVE	ACTIVE
QUEUED			1	
DEFERRED			2	
INACTIVE		2		3
ACTIVE			3	

1 : Queued Job Monitor

2 : Deferred Job Monitor

3 : Running Job Monitor

The amount known about a job in the various states and the disposition of various resources is summarized in the following table.

4.0 SCHEDULING
4.3 JOB STATES

STATE	RESOURCE WORKING SET	SYSTEM TABLES	CONTROL POINTS	NON-PREEMPTIBLE
QUEUED	none	*	none	none
DEFERRED	swap file	swap file	swap file	allocated
INACTIVE	swap file	in memory	in memory	allocated
ACTIVE	in memory	in memory	on dis-patch chain	allocated

* Known Job List Entry and Job Control Block allocated.

4.4 SCHEDULER TABLES

As mentioned earlier, the schedulers are taole driven. This section describes the various tables used by the schedulers to determine their decision making process. The method of supplying these tables initially is yet to be determined. Requests are provided to modify the tables while the system is running.

4.4.1 CLASS ATTRIBUTE TABLE

The Class Attribute Table (CAT) defines the attributes and characteristics of each class of jobs. There can be any number of classes (installation defined) but there must be at least three: System, Initial Batch, and Initial Interactive. These are the classes that the system job is in and batch and interactive jobs are in respectively when they are first established.

CLASS ATTRIBUTE TABLE

	SYSTEM	INITIAL BATCH	INITIAL INTER-ACTIVE	INSTALLATION DEFINED		
ATTRIBUTE	0	1	2	3	...	n
CPL Non-preempt						
CPL Deferred						
CPL Running						
Maximum Priority						
Maximum Known Jobs						
Maximum Running Jobs						
Maximum Active Jobs						
Maximum Total Working Set						
Major Time Slice						
Minor Time Slice						
Dispatch Time Slice						
PFF Goal						
Lowest Scheduling Level						
Non-preempt Priority Increment						
Deferred Priority Increment						
Inactive Priority Increment						
Active Wait Time						
Inactive Wait Time						
Maximum Wait Time						
Class Residence Time						
Next Class (Time)						
Maximum Job Working Set						
Next Class (WS)						

CPL Non-preempt:
CPL Deferred:
CPL Running:

CPL = Class Priority Level
These fields define the priority relationship among classes for different phases of scheduling. There may be as many levels as classes but there can be fewer; e.g., several classes may have the same CPL for non-preemptible resources.

CPL is an integer 0..n with 0 having the most priority and n the least. This is so that classes may be added with little impact on previously defined classes.

Maximum Priority:

The highest base priority that a job in this class can have.

Maximum Known Jobs:
Maximum Running Jobs:
Maximum Active Jobs:

The maximum number of jobs of this class which can be simultaneously known to the system or in the Running and Active states, respectively.

Maximum Working Set:

Maximum real memory simultaneously available to all jobs in this class.

Major Time Slice:
Minor Time Slice:
Dispatch Time Slice:

The time slices to be allocated jobs in this class when they go to the Running State, Active State, or one of their control points is dispatched on a processor, respectively.

PFF Goal:

Page Fault Frequency goal for jobs in this class. The schedulers will adjust running parameters to normalize a job's fault frequency to this value.

Lowest Scheduling Level:

Limits jobs in this class to the Deferred_Running_Active, Running_Active, or Active only states.

Deferred Priority Increment:
Inactive Priority Increment:
Non-preempt Priority
Increment:

The amount to increment a job's effective priority based on time job has been in the Deferred or Inactive states respectively or requires non-preemptible resources.

Active Wait Time:
Inactive Wait Time:

The length of time that all control points of a job must be waiting before it should be moved from the Active to Inactive or Inactive to Deferred states respectively.

Maximum Wait Time: The maximum amount of time jobs of this class can wait before it is considered an error state.

Class Residence Time: Amount of CPU time a job can stay in this class before it is automatically switched to another class.

Next Class (Time): The class to which to switch the job if its Class Residence Time has expired.

Maximum Job Working Set: The maximum working set a job can have before it is automatically switched to another class.

Next Class (WS): The class to which to switch the job if its maximum jobs working set limits has ben reached.

4.4.2 CLASS PRIORITY LEVEL TABLE

The Class Priority Level Table (CPLT) defines the decisions and algorithms used by the various schedulers to perform the state transitions described in Section 4.3. There is an entry for each CPL used in the CAT. There are parameters in each entry about each state transition plus parameters controlling the method of allocation of non-preemptible resources.

CLASS PRIORITY LEVEL TABLE (CPLT)

Class Priority Level	0	1	...	M
Deferred to Inactive Transition				
Method				
Priority Cutoff				
Termination				
Inactive to Deferred Transition				
Inactive to Active Transition				
Method				
Priority Cutoff				
Termination				
Active to Inactive Transition				
Method				
Non-preement Resource Allocation				
Method				
Priority Cutoff				
Termination				
Partial Allocation				

Class Priority Level: The pertinent class priority level (CPL) of the entry. The number of priority levels is smaller than or equal to the number of classes.

Deferred to Inactive State Transitions: The parameters in this section direct the Deferred Job Monitor when it selects a deferred job to make running.

Method; Round-Robin

Jobs are examined on a circular basis. If a job cannot be made running for some reason, the next job in the CPL is examined until all jobs in the CPL have been examined.

Method; FIFO

Jobs are made running in the order they appeared in the deferred state. If a job cannot be made running, no other jobs in the CPL are examined.

Method; Priority

Jobs are examined in effective priority order. If a job cannot be made running for some reason, the next lower effective priority job is examined until all jobs in the CPL have been examined.

Priority Cutoff:

If the Method; Priority was selected, then jobs with an effective priority higher than the priority cutoff value will be treated as if Method; FIFO were selected.

Termination:

Determines whether or not to examine jobs in the next lower CPL if there are remaining deferred jobs in this CPL.

Inactive to Deferred Transition:

There are no parameters for this transition. Jobs move from inactive to deferred based on Inactive Wait Time and Major Time Slice from the CAT.

Inactive to Active State Transitions:

The parameters in this section direct the Running Job Monitor when it selects an inactive job to make active.

Method; Round-Robin

See Deferred to Inactive Transition

Method; FIFO

See Deferred to Inactive Transition

Method; Priority

See Deferred to Inactive Transition

Priority Cutoff:

See Deferred to Inactive Transition

Termination:

See Deferred to Inactive Transition

Active to Inactive Transition:

Normally jobs move from active to inactive based on Minor Time Slice and Active Wait Time. In some circumstances, however, it is necessary for the scheduler to deactivate a job for other reasons; e.g., system thrashing. This parameter specifies the method used to select a job to deactivate.

Priority:

Deactivate the job of lowest priority in this CPL.

Least Time Slice:

Deactivate the job which has the least remaining Minor Time Slice.

System Fit:

Deactivate the job which will do most to alleviate the situation which requires a job to be deactivated.

None:

Do not forcibly deactivate jobs in this CPL.

Non-preemptible Resource Allocation:

The parameters in this section direct the Resource Scheduler when it selects a job to which to allocate non-preemptible resources.

Method; Round-Robin

See Deferred to Inactive Transition

Method; FIFO

See Deferred to Inactive

	Transition
Method; Priority	See Deferred to Inactive Transition
Priority Cutoff	See Deferred to Inactive Transition
Termination	See Deferred to Inactive Transition
Partial Allocation	Indicates whether or not some of a jobs non-preemptible resource requests should be satisfied if all of them cannot be.

4.4.3 CLASS TRANSITION TABLE

The Class Transition Table (CTT) specifies which class changes are valid when requested by the JM#SET_CLASS request.

CLASS TRANSITION TABLE (CTT)

	DESTINATION CLASS				
ORIGINAL CLASS	SYSTEM	INITIAL BATCH	INITIAL INTERACTIVE	...	N
SYSTEM	Y	Y	Y	Y	Y
INITIAL BATCH	N	Y	N	*	*
INITIAL INTERACTIVE	N	N	Y	*	*
. . .	N	*	*	*	*
N	N	*	*	*	Y

* Installation Specified

4.4.4 SCHEDULER CONTROL TABLE (SCT)

The Scheduler Control Table contains some parameters which define the general operating characteristics of the various schedulers.

Frequency of Queued Job Monitor:
Frequency of Deferred Job Monitor:
Frequency of Running Job Monitor: The frequency of activation of these schedulers. It is assumed that the QJM should be at least an order of magnitude less frequent than the DJM, which should be at least an order of magnitude less frequent than the RJM.

PFF Floor:
PFF Ceiling: The schedulers will attempt to maintain the total page fault frequency in the system somewhere in the range PFF Floor < System PFF < PFF Ceiling.

I/O Rate Floor:
I/O Rate Ceiling: The schedulers will attempt to maintain the total frequency of I/O requests in the system in the range I/O Rate Floor < System I/O Rate < I/O Rate Ceiling.

Number of Classes: The number of classes specified in the CAT and CTT.

Highest CPL: The highest CPL specified in a CAT entry.

5.0 ENVIRONMENT OF STANDARD JOBS

Certain facilities/properties are provided by the system as constituents of all standard jobs; those constituents of which the user should be cognizant are are discussed in this section.

5.1 ADDRESS SPACE

Segment 5: The System Global Logical Name Space (SGLNS) segment; this global segment is interpretively readable by the user.

Segment 6: The User Local Logical Name Space (ULLNS) segment for the job; the segment is directly and interpretively readable and writable by the user.

Segment 7: The System Local Logical Name Space (SLLNS) segment for the job; the segment is interpretively readable and writable by the user.

Segment 8: The User Global Logical Name Space (UGLNS) segment; this global segment is interpretively readable and writable by the user.

5.2 FILES

JOB#INPUT: The logical name unconditionally assigned to the Job's Primary Input File; this file is for the exclusive use of Command Language Interpreter and serves as the default source of command language statements in the Job.

JOB#DAYFILE: The logical name assigned to a sequentially organized, disk resident file which is used to record sequential history of activities in the Job (e.g., images of command language statements processed, messages to/from operators, user supplied information). This file may be written by system elements in a Job as well as by the user via record level I/O functions. This file is routed according to the job's Primal Invocator Identifier (PII) by

Job Terminator in the absence of any user specified DIRECT specification.

JOB#PRINT The logical name assigned to a sequentially organized disk resident file which is unconditionally provided by the system. This file is provided as a default file to be used, via record level I/O, by elements of the system and the product set (e.g., loader maps, compiler source listing). In the absence of any user specified DIRECT specification, this file is routed by Job Terminator according to the Job's PII.

JOB#OUTPUT: The logical name of a file which is used by Command Language Interpreter for communications directed to the user (e.g., command language functional, syntactical or parameter error notification). In a batch Job JOB#OUTPUT is an alias logical name for JOB#DAYFILE, i.e., Command Language Interpreter error notifications transmitted to JOB#OUTPUT actually are recorded on the physical file associated with JOB#DAYFILE. For an interactive Job JOB#OUTPUT is the logical name of the output side (e.g., display) of the interactive device and, consequently, Command Language Interpreter error notifications are transmitted to the interactive user; such messages are also transmitted to JOB#DAYFILE by Command Language Interpreter. JOB#OUTPUT is written using record level I/O functions.

5.3 OTHER

JOB#JCB: An LNS alias through which the user, and system elements within the Job address space, may reference the Job Control Block of the Job. JOB#JCB resides in the Job's SLLNS segment.

6.0 JOB MANAGEMENT REQUESTS

6.1 REQUEST PURPOSE AND PARAMETER DESCRIPTION

The following requests are available for communication with the various Job Management facilities.

6.1.1 JM#SUBMIT

The JM#SUBMIT request is utlized to request the creation of an asynchronously executing job in a new address space.

JM#SUBMIT (fname, jname, eo, do, status)

fname : The name of a File Control Block which resides in a logical name space (LNS), and which describes the file to be used as the Primary Input File for the job being invoked; specification of this parameter is mandatory. The file must be permanent, and an ACL defined for it to permit access under the account and user identifiers under which the new job will run.

jname : Specification of this parameter is optional in user jobs; if specified, it is interpreted to be the name of an alias variable which the system will declare in the user job's System Local Logical Name Space (SLLNS); the user may reference fields in the Job Control Block of the job being invoked by referring to the name of the SLLNS alias variable.

Specification of this parameter by the system programs Stager and System Access Manager is mandatory ; in these instances, the parameter is interpreted to be the name of a Job Control Block previously declared in System Global Logical Name Space (SGLNS).

eo : This parameter value is used to specify whether immediate establishment is desired, or delayed establishment is permissible. One of the following values for eo may be specified:

Q : Delayed establishment is permissible
null : same as Q
I : Reject if immediate establishment is not possible

do : This parameter value is used to specify the desired disposition of the Primary Input File following the running of the job being invoked. One of the following values may be specified for do:

P : Purge the file when job terminates
K : Do not purge the file
null : same as K

The P option will be honored only when the user identifier of the terminating Job matches the owner identifier of the permanent file.

status: returned request status.

6.1.2 JM#DIRECT

The JM#DIRECT request permits the specification of explicit parameters for use with a file during the subsequent issuance of a JM#ROUTE request with some or all of the parameters omitted. Any optional parameters supplied during a JM#ROUTE request will override the corresponding explicit parameters from a JM#DIRECT request.

JM#DIRECT (file, destination, form, copies, status)

file: The logical name of the file (temporary of permanent) to be directed.

destination: An optional parameter which specifies the logical destination name of the location to which to direct the file. If not specified, the default destination from the JM#ROUTE request is used.

form: An optional parameter which specifies the physical properties of the output medium (e.g., paper size, card type, printer train, ribbon color) upon which the data is to be placed. If not specified, the default form from the

JM#ROUTE request is used.

copies: An optional parameter which specifies the number of copies to be placed on the output unit. If not specified, the default number from the JM#ROUTE request is used.

status: returned request status.

6.1.3 JM#RETRACT

The JM#RETRACT request is used to cancel a previously issued JM#DIRECT request on the same file. The entry that was placed in the local directed file list for the specified file is removed.

JM#RETRACT (file, status)

file: The file to be retracted.

status: returned request status.

6.1.4 JM#ROUTE

The purpose of the JM#ROUTE request is to initiate the transmittal of a file (temporary or permanent) to some destination. If no explicit value is given for an optional parameter, the explicit values from a previously issued JM#DIRECT request on that logical file name will take precedence over the defualt values in effect at the time the JM#ROUTE request is issued.

JM#ROUTE (file, destination, form, copies, status)

file: The logical name of the file (temporary or permanent) to be transmitted.

destination: An optional parameter which specifies the destination to which to direct the file. If not specified, the default destination is assumed.

form: An optional parameter which specifies the physical properties of the output medium (e.g., paper size, card type, printer train, ribbon color) upon which the data is

to be placed. If not specified, the default medium is assumed.

copies: An optional parameter which specifies the number of copies to be placed on the output unit. If not specified, the default number of copies is assumed.

status: returned request status.

6.1.5 JM#SYSLOG

The JM#SYSLOG request is utilized by elements of IPLOS to effect the transfer of information to the SYSTEM LOG FILE.

JM#SYSLOG (recid, addr, bytes, status)

recid: A SYSTEM LOG FILE record identifier code of the form tt/cc/ss/ (refer to section on System Logging for a discussion of Log File Record Identifier.

addr: The PVA of the beginning of the information which is to be transferred to the SYSTEM LOG FILE. Information which begins at the specified address must be readable within the addressing context of the job in which the JM#SYSLOG request is issued.

bytes: The number of bytes of information to be transferred to the SYSTEM LOG FILE.

status: returned request status.

Specification of all parameters is required.

6.1.6 JM#CLAIM

The purpose of this request is to establish the maximum simultaneous usage of a class of peripheral devices by a job. A claim must be established for a class of devices before a JM#RESERVE request may be issued for devices of the class. Claims may not be increased while any devices are allocated to a job. Claims may be decreased at any time down to the number of devices currently allocated to the job.

JM#CLAIM (unit_class, number, status)

unit_class: the class of devices for which a claim is being made.

number: the number of devices of the class being claimed.

status: returned request status.

6.1.6.1 Logic

JM#CLAIM Request Processor Entry

Valid Unit Class	No	Yes				
Claim increase		Yes			No	
Any allocated units		Yes	No			
Claim exceed limits			Yes	No		
Less than allocated					Yes	No
Error status	X	X	X		X	
Enter claim				X		X
Signal Scheduler				X		X
Return	X	X	X	X	X	X

6.1.7 JM#CHANGE_CLAIM

The purpose of this request is to increase or decrease the number of units of a class previously claimed through JM#CLAIM or JM#CHANGE_CLAIM request. Claims may not be increased while any devices are attached to the job. Claims may be decreased at any time down to the number of devices currently allocated to the job.

JM#CHANGE_CLAIM (unit_class, number, status)

unit_class: the class of devices for which the claim is being modified.

number: the signed value to be added to the existing claim.

status: returned request status.

6.1.7.1 Logic

JM#CHANGE_CLAIM Request Processor Entry

Valid Unit Class	No	Yes				
Claim increase		Yes			No	
Any allocated units		Yes	No			
Resultant claim exceed limit			Yes	No		
Less than allocated					Yes	No
Error status	X	X	X		X	
Adjust claim				X		X
Signal scheduler				X		X
Return	X	X	X	X	X	X

6.1.8 JM#RESERVE

The purpose of this request is to register the requirement for a non-preemptible resource (file, volume set, or unit set. The allocation of the resource to the job will not take place until the issuance of a JM#ACQUIRE request. This request permits groups of resources to be acquired simultaneously. This reduces the possibility of deadlock and increases total system efficiency by not having some resources attached to a job while it still needs others to continue execution.

JM#RESERVE (resource, usage, status)

resource: an (LNS) structure defining the resource to be reserved. Allowable types are File Control Block, Volume Set Control Block and Unit Set Control Block.

usage: indicates whether the resource is to be obtained for shared use or private to the job.

status: returned request status.

6.1.8.1 Logic

JM#RESERVE

Request processor entry point

JM#ACQUIRE outstanding	Yes	No			
Resource a FCB		Yes	No		
Resource a VSCB			Yes	No	
Resource a USCB				Yes	No
Error	X				X
Call RESERVEF		X			
Call RESERVEV			X		
Call RESERVEU				X	
Return	X	X	X	X	X

RESERVEF Procedure

File reserved or attached	No						Yes
Permanent file	No	Yes					
VSCB named		No	Yes				
VS reserved or attached			No		Yes		
VS containing catalogue on-line			No	Yes	No	Yes	
To be supplied	X						
Supply system V.S. name		X					
Call JM#RESERVE (VSCB)			X	X			
Mark FCB as Reserved		X	X	X	X	X	
Get catalogue entry		X		X		X	
Build File Reservation Record		X	X	X	X	X	
File unique name in FRR		X		X	X		
File external name in FRR			X		X		
Error status							X
Return	X	X	X	X	X	X	X

RESERVEV Procedure

Volume set reserved or attached	Yes	No							
Are volumes specified		Y				N			
Is USCB named		Y			N	Y			N
Is other VSCB associated with USCB		Y	N			Y	N		
Is USCB reserved or attached			Y	N			Y	N	
Error status	X	X				X			
Get catalogue entry							X	X	X
Build USCB					X				X
Call JM#RESERVE				X	X			X	X
Build VS reservation record			X	X	X		X	X	X
Mark VSCB as reserved			X	X	X		X	X	X
Return	X	X	X	X	X		X	X	X

RESERVEU Procedure

Unit set reserved or attached	Yes	No		
Exceed CLAIM		Yes	No	
VS associated with USCB			Yes	No
Error status	X	X		
Build Unit reservation record				X
Mark USCB as reserved			X	X
Return	X	X	X	X

6.1.9 JM#CANCEL_RESERVE

The purpose of this request is to cancel the effects of all JM#RESERVE requests issued since the last JM#ACQUIRE request.

JM#CANCEL_RESERVE (status)

status: returned request status.

6.1.9.1 Logic

JM#CANCFL_RESERVE

Request processor entry

JM#ACQUIRE outstanding	Yes	No					
Any more resource reservation records		No	Yes				
File reservation record			Yes	No			
Volume set record				Yes	No		
Unit set record					Yes		No
System supply USCB					Yes	No	
Error status	X						X
Mark FCB as not reserved			X				
Mark VSCB as not reserved				X			
Mark USCB as not reserved						X	
Return USCB					X		
Return reservation record			X	X	X	X	
Return	X	X	X	X	X	X	X

6.1.10 JM#ACQUIRE

The purpose of this request is to cause the system to satisfy all requests previously established through the JM#RESERVE request. All outstanding requests must be satisfied before any additional JM#RESERVE or JM#ACQUIRE requests can be issued.

JM#ACQUIRE (queue, [ecb], status)

queue: indicates whether the request for resources should be queued if they cannot all be satisfied immediately or the JM#ACQUIRE be terminated with none of the allocations made.

ecb: optional parameter which names an event to be PM#CAUSED when all outstanding requests have been satisfied. When the "queue" parameter indicates immediate termination, the specification of an ecb is not permitted and if present will result in an error status.

status: returned request status.

6.1.10.1 Logic

JM#ACQUIRE

Request Processor Entry

Is JM#ACQUIRE outstanding	Yes	No						
Resource Reservation Records		No			Yes			
ECB named		No	Yes		Yes		No	
Immediate Return			No	Yes	Yes	No	Yes	No
Error Status	X			X	X			
Select Reply no trap							X	X
Select Reply trap to Allocate						X		
Signal Resource Allocate						X	X	X
Wait for reply							X	X
Call ALLOCATE							X	X
Cause ECB			X					
Return	X	X	X	X	X	X	X	X

ALLOCATE: Procedure called by JM#ACQUIRE or via trap on reply from Resource Allocator.

Any more Resource Res Recs	No		Yes			
File Request			Yes	No		
Volume Set Request				Yes	No	
Unit Set Request					Yes	No
ECB named in JM#ACQUIRE	No	Yes				
Call ATTACHF			X			
Mark VSCB as attached				X		
Mark USCB as attached					X	
Remove Resource Res. Rec.			X	X	X	
Pick next RRR			X	X	X	
Loop to ALLOCATE			X	X	X	
Cause ECB		X				
Return	X	X				X
Error Status						X

6.1.11 JM#RETURN

The purpose of this request is to return a file volume set or unit set previously allocated to a job through JM#RESERVE/JM#ACQUIRE commands. If a file is named, then only the file is returned. If a volume set is named then all files on the volume set that are attached to the job and the volume set are returned. If, in this case, the unit set was supplied by the system, it is also returned. If the unit set was supplied by the job, it is left allocated to the job. If a unit set is named, the attached files, volume set and unit set are returned.

JM#RETURN (resource, [claim_disposition], status)

resource: the (LNS) structure defining the resource to be returned. Allowable types are File Control Block, Volume Set Control Block, and Unit Set Control Block.

claim_disposition: if the unit set involved is returned, indicates whether or not to reduce the claim for the appropriate device class.

status: returned request status.

6.1.11.1 Logic

JM#RETURN

Request processor entry point

Is JM#ACQUIRE outstanding	Yes	No			
Is resource a FCB		Yes	No		
Is resource a VSCB			Yes	No	
Is resource a USCB				Yes	No
Error status	X				X
Call RETURNF		X			
Call RETURNV			X		
Call RETURNU				X	
Return	X	X	X	X	X

RETURNF procedure

Is File attached	Yes	No
Call DETACHF	X	
Signal Resource Allocator	X	
Error status		X
Return	X	X

RETURNV Procedure

Is VSCB allocated	No	Yes					
Any more attached files		Yes	No				
Is volume set shareable			Yes		No		
Called by RETURNU			Yes	No	Yes	No	
Was unit set supplied by system						Yes	No
Error status	X						
Pick first attached FCB		X					
Call RETURNF		X					
Loop to RETURNV		X					
Mark USCB not reserved			X	X	X	X	X
Signal resource allocator			X	X	X	X	X
Call RETURNU				X		X	
Return	X		X	X	X	X	X

RETURNU Procedure

Is USCB allocated	No	Yes			
Is VSCB attached		Yes		No	
Is claim to be reduced		Yes	No	Yes	No
Error Status	X				
Mark USCB not attached		X	X	X	X
Call RETURNV		X	X		
Signal resource allocator		X	X	X	X
Reduce number claimed		X		X	
Return	X	X	X	X	X

6.1.12 JM#SET_PRIORITY

The purpose of this request is to establish or change the base priority of a job.

JM#SET_PRIORITY (job,priority,status)

job: LNS descriptor of the Job Control Block or alias of the job to have its priority changed.

priority: The value of 1..15 to set the priority.

status: returned request status.

Example: JM#SET_PRIORITY (this_job,'7',status);

6.1.13 JM#SET_CLASS

The purpose of this request is to change the class of a job.

JM#SET_CLASS (job,class,status)

job: LNS descriptor of the Job Control Block or alias of the job to have its class changed.

class: The value (1..n) where n is the highest class number to set the class.

status: returned request status.

Example: JM#SET_CLASS (this_job,xaction,status);

6.1.14 JM#GET_CAT

The purpose of this request is to get a copy of a record of the Class Attribute Table.

JM#GET_CAT (class,buffer,status)

class: The class number of the record desired.

buffer: The location to put the copy of the record.

status: returned request status.

6.1.15 JM#REPLACE_CAT

The purpose of this request is to replace a Class Attribute Table record with a new one.

JM#REPLACE_CAT (class,buffer,status)

class: The Class number of the record to be replaced.

buffer: Location of the new record.

status: returned request status.

6.1.16 JM#GET_CPLT

The purpose of this request is to get a copy of a Class Priority Level Table record.

JM#GET_CPLT (cpl,buffer,status)

CPL: Class priority level of the desired record.

buffer: Location into which to place the record.

status: returned request status.

6.1.17 JM#REPLACE_CPLT

The purpose of this request is to replace a Class Priority Level Table record with a new one.

JM#REPLACE_CPLT (cpl,buffer,status)

CPL: Class priority level of the record to be replaced.

buffer: Location of the new record.

status: returned request status.

6.1.18 JM#GET_CTT

The purpose of this request is to get a copy of a class Transition Table record.

JM#GET_CTT (class,buffer,status)

class: The class number of the desired record.

buffer: The buffer into which to place the record.

status: returned request status.

6.1.19 JM#REPLACE_CTT

The purpose of this request is to replace a Class Transition Table record with a new one.

JM#REPLACE_CTT (class,buffer,status)

Class: The class of the record to be replaced.

buffer: Location of the new record.

status: returned request status.

6.1.20 JM#GET_SCT

The purpose of this request is to get a copy of the Scheduler Control Table.

JM#GET_SCT (buffer,status)

buffer: Location to place the SCT.

status: returned request status.

6.1.21 JM#REPLACE_SCT

The purpose of this request is to replace the current Scheduler Control Table with a new one.

JM#REPLACE_SCT (buffer,status)

buffer: The location of the new SCT.

status: returned request status.

6.2 REQUEST STATUS

Each terminating request processor returns status information, in a coded format, which describes the disposition of the request.

6.2.1 DEFINITION OF STATUS CODES

Accepted

0 JM 000	accepted and completed
4 JM 000	accepted and queued for service

Rejected

Parameter Error Rejects

8 JM 101	required parameter not specified
8 JM 102	invalid parameter type
8 JM 103	invalid parameter value
8 JM 104	undefined LNS variable name
8 JM 105	duplicate LNS variable name
8 JM 106	invalid address specifled
8 JM 107	(undefined status code)
•	•
•	•
•	•
8 JM 1FF	(undefined status code)

Functional Error Rejects

8 JM 201	attempted to submit interactive job
8 JM 202	disk file not permanent
8 JM 203	(undefined status code)
•	•

•	•
•	•
8 JM 2FF	(undefined status code)

Internal Condition Rejects

C JM 901	system limits exceeded
C JM 902	no table space available
C JM 903	(undefined status code)
•	•
•	•
•	•
C JM 9FF	(undefined status code)

Internal Error Rejects

F JM A01	LNS variable not in proper LNS segment
F JM A02	(undefined status code)
•	•
•	•
•	•
F JM AFF	(undefined status code)

6.2.2 STATUS CODES BY REQUEST

	SUBMIT	SYSLOG	DIRECT	ROUTE	SET-PRIORITY	SET-CLASS	GET-CAT	REPLACE-CAT	GET-CPLT	REPLACE-CPLT	GET-CTT	REPLACE-CTT	GET-SCT	REPLACE-SCT	CLAIM	CHANGE-CLAIM	RESERVE	RETURN	ACQUIRE	CANCEL-RESERVE
0 JM 000	X	X																		
4 JM 000	X																			
8 JM 101	X	X																		
8 JM 102																				
8 JM 103		X																		
8 JM 104	X																			
8 JM 105	X																			
8 JM 106		X																		
8 JM 201	X																			
8 JM 202	X																			
C JM 901	X																			
C JM 902																				
F JM A01	X																			

7.0 PROCESS DESCRIPTIONS

7.1 SUBMIT REQUEST PROCESSOR

To be supplied

7.1.1 DECISION TABLES

Is a FILE = <fname> parameter specified	Y														N
Does FCB = <fname> reside in LNS	Y													N	
Does FCB = <fname> describe disk or interactive file	Y												N		
Is Alpha the System Job	Y							N							
Does FCB = <fname describe a disk resident file	N			Y				Y				N			
Is the disk resident file permanent				Y			N	Y			N				
Is a JOB = <jname> parameter specified	Y		N	Y		N		N	Y						
Does JCB = <jname> reside in SGLNS	Y	N		Y	N										
Does Alias = <jname> reside in SLLNS									N	Y					
1. Set return status : Reject; FILE parameter absent															X
2. Set return status : Reject; FCB = <fname> undefined														X	
3. Set return status : Reject; Illegal input file medium													X		
4. Set return status : Reject; Input file not on disk												X			
5. Set return status : Reject; Disk file not permanent							X				X				
6. Set return status : Reject; <jname> not unique										X					
7. Set return status : Reject; JOB parameter absent			X			X									
8. Set return status : Reject; JCB = <jname> undefined		X			X										
9. Declare a JCB = <unique> in SGLNS								X	X						
10. Declare an Alias = <jname> in SLLNS									X						
11. Set JCB name into the Alias variable									X						
* 12. Copy User id. from JCB.ALPHA to JCB.BETA								X	X						
13. Copy Account id. from JCB.ALPHA to JCB.BETA								X	X						
14. Copy PII from JCB.ALPHA to JCB.BETA								X	X						
15. Set Job Type field of JCB.BETA to "User Job"								X	X						
16. Copy Initial Priority from JCB.ALPHA to JCB.BETA								X	X						
17. Set Job Class field of JCB.BETA to "batch"				X				X	X						
18. Set Job Class field of JCB.BETA to "interactive"	X														
19. Set name of JCB into JCB.BETA	X			X				X	X						
20. Set current date/time into JCB.BETA	X			X				X	X						
21. Set Job Status field of JCB.BETA to "known/not estab."	X			X				X	X						
22. Obtain perm. file name from FCB - place in JCB.BETA	X			X				X	X						
23. Set ENQ field of JCB.BETA with ENQ parameter value	X			X				X	X						
24. Set DISP field of JCB.BETA with DISP parameter value	X			X				X	X						
25. Send SIGNAL containing JCB.BETA name to Job Estab.	X			X				X	X						
26. Wait for reply signal from Job Establisher	X			X				X	X						
27. Proceed with Phase II when reply signal arrives	X			X				X	X						
28. RETURN immediately to caller {with return status}		X	X		X	X	X			X	X	X	X	X	X

* ALPHA is used here to designate entities associated with the submittor Job; BETA designates entities associated with the Job being submitted.

SUBMIT REQUEST PROCESSOR - PHASE I

Is Alpha the System Job	Y		N			
Does status from Job Establisher indicate "reject"	Y	N	Y		N	
Did Phase I create an Alias of <jname> in SLLNS			N	Y	N	Y
1. Link JCB.BETA into ALPHA's Submit thread						X
2. REMOVE SLLNS Alias = <jname>				X		
3. REMOVE JCB.BETA previously declared by Phase I			X	X		
4. Set return status : {That returned by Job Establisher}	X	X	X	X	X	X
5. RETURN to caller with return status	X	X	X	X	X	X

SUBMIT REQUEST PROCESSOR - PHASE II

7.2 JOB ESTABLISHER TASK

To be supplied

7.2.1 DECISION TABLE

Is the signal-specified JCB in SGLNS	Y								N
Was a KJL entry previously assigned (i.e., queued)	N						Y		
Is a KJL entry available for assignment	Y					N			
Does the JCB specify an interactive job	Y		N						
Is I.crnt<I.max	Y	N							
Is B.crnt<B.max			Y	N			Y	N	
Does the JCB specify queue permission				Y	N				
1. Set return status : Reject; JCB =<name> undefined									X
2. Set return status : Reject; KJL limits exceeded						X			
3. Set return status : Reject; Interactive limits exceeded		X							
4. Set return status : Reject; Batch limits exceeded					X				
5. Acquire a KJL entry	X		X	X					
6. Inter-link JCB and KJL	X		X	X					
7. Set KJL entry status : Queued for Establishment				X					
8. Set KJL entry priority from JCB				X					
9. Set JCB status field : Known/not established				X					
10. Set return status : Accepted and Queued				X				X	
11. Issue CREATE_ADDRESS_SPACE	X		X				X		
12. Set KJL entry status : Establish - swapped in	X		X				X		
13. Set JCB status field : Est. - not initiated	X		X				X		
14. Trigger initial execution of SEQ.MTR in new Job	X		X				X		
15. Set I.crnt = I.crnt + 1	X								
16. Set B.crnt = B.crnt + 1			X				X		
17. Set return status : Accepted, established	X		X				X		
18. Send reply signal with return status to requestor	X	X	X	X	X	X	X	X	X

JOB ESTABLISHER TASK

7.3 QUEUED JOB MONITOR

To be supplied

7.3.1 DECISION TABLES

	Y	N
A. Is the length of KJL "queue" thread ≠ 0		
1. Set index {i} to first KJL "queue" thread entry	X	
2. Set index {j} to {{i}. next}	X	
3. Proceed with Phase II	X	
4. Wait on time event		X
5. Proceed at {A} when Wait is satisfied		X

QUEUED JOB MONITOR - PHASE I

	Y	N	
		Y	N
A. Is {j} = 0 Is {{j}.priority}>{{i}.priority}			
1. Set {i} = {j}		X	
2. Set {j} = {{j}.next}		X	X
3. Proceed at {A}		X	X
4. Extract {{i}.jcbid}	X		
5. Send signal to Job Establisher	X		
6. Wait for reply signal from Job Establisher	X		
7. Proceed with Phase III when Wait is satisfied	X		

QUEUED JOB MONTIOR - PHASE II

Was establishment request accepted	Y		N
Was the Job established	Y	N	
Wait on time event		X	
Transmit a message to System Error stream			X
Advise System Operator of the error			X
Wait for System Operator's acknowledgement			X
Proceed with Phase I	X	X	X

QUEUED JOB MONITOR - PHASE III

7.4 SYSLOG REQUEST PROCESSOR

To be supplied.

7.4.1 DECISION TABLE

Are RECID, LOC, and LENGTH parameters specified	Y				N
Is TYPE field of RECID parameter valid	Y			N	
Is address specified for LOC valid	Y		N		
Is value specified for LENGTH valid	Y	N			
Set return status: Reject; required parameter absent					X
Set return status: Reject; invalid parameter value		X		X	
Set return.status: Reject; invalid memory address			X		
Acquire contiguous memory sufficient for record	X				
header and text in a global segment	X				
Construct record header in acquired global memory	X				
Append text to header in global memory	X				
* Send signal to LOGTASK in System Job	X				
Set return status: Accepted and transmitted	X				
Return to requestor with status	X	X	X	X	X

* Signal includes address of the record in the global segment and the length of the record

JM*SYSLOG REQUEST PROC.

7.5 DIRECT REQUEST PROCESSOR

The JM#DIRECT request is one of the Task Services in the user job. The request is processed to completion before control is returned to the requestor. Note that the JM#DIRECT request does not initiate routing; it only associates a destination with a file. The issuance of a subsequent JM#DIRECT request or a JM#ROUTE with explicit parameters will take precedence over a previous JM#DIRECT request.

During the initilization of every job, a local directed file list will be provided in the local LNS segment. Whenever a JM#DIRECT request is issued, an entry containing the file along with the explicit parameters that were specified on the request will be placed in the local directed file list. When a JM#ROUTE request is issued, the local directed file list will be consulted. If the file appears, the explicit parameters in the entry will be used in lieu of the default values for the JM#ROUTE request. That entry in the local directed file list will then be removed.

If a subsequent JM#DIRECT request is issued for the same file, all informatiom from the former request will be discarded.

The destination will probably be the name of an output queue. During (and after) the autoload sequence for the system, a number of standard output queues will be declared in Global LNS; they will then be ONSYSTEMed to one or more output units. Subsequently, when a file is placed in a given output queue, the contents will be transferred to the appropriate output unit.

Note that the above paragraph does not preclude the use of an alias for the output queue. In fact, the use of an alias for the destination will likely be the normal mode of operation since the presence of that mechanism allows for the use of universal destination names such as PRINTER and PUNCH. The system profile would then provide the relationship between these names and the output queues. Thus, a central site which served many users would need only one queue for each of the standard default destinations.

Additional output queues would be needed for each additional unique type of output unit. Then, if a particular user was temporarily offsite or needed to use a non-standard output unit, it would be an extremely simple matter for him to redefine the alias in order to direct the file to a different unit without any changes in the actual programs.

Tables Required

Local Directed File List

A Local Directed File List (LDFL) will have entries which contain the following information:

- o File - the descriptor of the fcb
- o Destination - the descriptor of the qcb
- o Form - The actual string of characters giving the form to be used
- o Copies - The actual number of copies

7.6 RETRACT REQUEST PROCESSOR

The JM#RETRACT request is one of the Task Services in the user job. The request is processed to completion before control is returned to the requestor.

During the initilization of every job, a local directed file list will be provided in the local LNS segment. Whenever a JM#DIRECT request is issued, an entry containing the file along with the explicit parameters that were specified on the request will be placed in the local directed file list. When a JM#RETRACT request is issued, the local directed file list will be consulted. If the file appears, that entry will be removed.

7.7 ROUTE REQUEST PROCESSOR

The JM#ROUTE request is one of the Task Services in the user job. The request is processed to completion before control is returned to the requestor. JM#ROUTE will attach the specified file to the output queue indicated by the specified destination.

The destination will probably be the name of an output queue. A number of standard output queues will be declared in Global LNS; they will then be ONSYSTEMed to one or more output units. Subsequently, when a file is placed in a given output queue, the contents will be transferred to the appropriate output unit.

Note that the above paragraph does not preclude the use of an alias for the output queue. In fact, the use of an alias for the destination will likely be the normal mode of operation since the presence of that mechanism allows for the use of universal destination names such as PRINTER and PUNCH. The system profile would then provide the relationship between these names and the output queues. Thus, a central site which served many users would need only one queue for each of the standard default destinations.

Additional output queues would be needed for each additional unique type of output unit. Then, if a particular user was temporarily offsite or needed to use a non-standard output unit, it would be an extremely simple matter for him to redefine the alias in order to route the file to a different unit without any changes in the actual programs.

It should also be noted that some sites may require a separation between the different users who wish to transfer data to the output unit. High priority files could be placed in one output queue while lower priority files go to a second or a third output queue. Storage space restrictions might require the operator to place all low priority files on backup storage and retrieve them for processing at a later time. The use of an alias which is available to both the system and the user makes such examples easy to handle.

When any explicit parameters in the JM#ROUTE request are omitted, a local directed file list for the job will be accessed to determine if a JM#DIRECT request is outstanding on the specified file. If one exists, the explicit values therein will be used to override all optional parameters left unspecified in the JM#ROUTE request, and the entry in the local directed file list for the specified file will be removed. Otherwise, standard default values are assumed which will be determined from entries in the local LNS segment.

If the file is temporary, JM#ROUTE will make the file permanent under a unique new name. However, JM#ROUTE will include in the output queue a directive to release the file after the file has been routed to its destination.

Tables Required

Output Queue

Each Output Queue entry will contain the following information:

o File	-	the descriptor of the fcb of the file
o Form	-	the actual string of characters
o Copies	-	the actual number of copies
o Job ID	-	the descriptor of the jcb of the job which submitted the ROUTE request

7.8 <u>FILE ROUTER</u>

File Router will periodically ascertain the state of all known output queues. When appropriate, File Router will activate and send to an instance of the specified Output Distributor its associated output queues. A set of tables will be used to control and direct the functions which File Router performs, i.e., File Router will be table driven.

Function Description

There will be only one File Router task in the System Job. Periodically, it will be invoked. Its sole function will be to check if there are any idle output units which are associated with one or more active, non-empty output queues. Whenever, it finds one in that condition, File Router will determine if the number of currently active Output Distributors allows an additional one to be SPAWNed. If that may be done, File Router will hand off to the new Output Distributor subtask the output unit and its associated set of output queues.

Tables Required

Known Output Queue List

Each entry in the Known Output Queue List will contain the following information:

o Queue	-	the descriptor of the qcb
o Status	-	the output queue may be either active or inactive
o Unit List	-	the descriptor of the list of output units to which the associated output queues are to be drained

Known Output Unit List

Each entry in the Known Output Unit List will contain the following information:

- o Unit - the descriptor of the ucb
- o Status - the output unit may be either active or inactive
- o Subtask - the descriptor of the stcb
- o File - the descriptor of the current file's fcb
- o Queue List - the descriptor of tne list of output queues from which the output unit is to be drained

7.8.1 FILE ROUTER DIRECTOR

File Router Director will be used to modify and change those tables which are used to control and direct the functions performed by File Router. This program will be available to system programs which have operator authorization level security to read and/or modify these tables via the standard system interface.

Function Description

Control and direction of the functions performed by File Router should normally require infrequent attention. In the usual mode of operation, the standard profile used by an operator's job will initialize the tables used by File Router. Thereafter, except for changes in printer forms, there will be few additional occasions (relative to the number of times that File Router is used) that File Router Director will be needed. Consequently, although the functions performed by this program are inherently part of File Router, they have purposely been divorced from that program. Synchronization locks will be required to ensure coordination between the two programs.

File Router Director will be able to receive commands from an operator communications program which may wish to regulate and/or modify the standard mode of operation for File Router. The following primitives will be accepted which may be used either singly or in combination to achieve the effect desired by

the operator's command.

(a) REGISTER (queue_list,unit_list,status)

Each output queue that File Router is to handle must be entered into the tables which control and direct the functions of File Router. An error status will be returned if an output queue does not exist or is already registered.

(b) REMOVE (queue_list,unit_list,status)

This primitive will remove the specified output queues from the tables used to control and direct the functions of File Router. If an output queue is active or is not presently registered, an error status will be returned.

(c) ACTIVATE (queue_list,unit_list,status)

This primitive will change the status of the specified queues from inactive to active. A number of the output queues, which are registered in the tables which control File Router, may be designated as inactive. Such output queues will retain all of their attributes, but will not be connected to an Output Distributor. If the output queue was already active or it was not registered with File Router, an error status will be returned.

(d) SUSPEND (queue_list,unit_list,status)

This primitive will change the status of the specified queues from active to inactive. A number of the output queues, which are registered in the tables which control File Router, may be designated as inactive. Such output queues will retain all of their attributes, but will not be dispatched to an Output Distributor. If the output queue was already inactive or it was not registered with File Router, an error status will be returned.

(e) ALTER (Output_Distributor,unit,status)

This primitive may be used to alter the Output Distributor associated with an output unit.

(f) LIST (queue,status)

to be supplied

(g) FORM (type,status)

to be supplied

7.8.2 OUTPUT DISTRIBUTOR

An Output Distributor will be used to transfer data from files which are attached to a set of output queues. When first given control, an output unit will also be specified to which the data will be transferred.

In addition, an Output Distributor will accept a number of primitive directives to control the processing of the files.

Function Description

There will be several different flavors of Output Distributor, one for each different type of output unit. (At present, three different types are known - printer, punch, 713 terminal.) Each unique Output Distributor may have several instances of itself active at any one time depending on the activity in the output queues, the mix of output units available, and the maximum number of instances allowed for all Output Distributors.

When an Output Distributor is SPAWNed, it will be given a set of output queues and an output unit. It will then proceed to transfer the data from the files in the output queues to the unit. The following steps outline the set of actions performed upon each file.

(a) Find the highest priority file attached to the given set of queues.

(b) Determine the file type.

(c) Determine the form type required.

(d) If the correct form is not on the unit, request the operator to change forms and wait for the reply.

(e) Select the Conversion Routine which performs the correct transfer of data from the file to the unit.

(f) Open the file.

(g) Transfer the file.

(h) Close the file.

(i) Release the file if that indication is present in the output queue record.

(j) Remove the output queue record for that file.

(k) Update the accounting log for the job which submitted the JM#ROUTE request to reflect the charges incurred to transfer the file to the output unit.

(l) Check the proceed indicators.

In addition to performing the basic function of directing the data from the file to the output unit, Output Distributor will be able to receive commands from an operator communication program which may wish to request and/or modify the standard mode of operation for any particular Output Distributor. The following primitives will be accepted which may be used either singly or in combination to achieve the effect desired by the operator's command:

(a) CONTINUE

This primitive will direct the Output Distributor to resume processing at the point where it is currently situated. If processing was interrupted in the middle of a file, and no other primitives have been issued, output to the unit will resume where it left off. If the Output Distributor is not in a hold mode, a null operation will result.

(b) HOLD

This primitive will direct the Output Distributor to cease processing. The input pointer will not be altered. However, all output buffers will be cleared. If the Output Distributor is already in a hold mode, a null operation will result.

(c) CANCEL

This primitive will direct the Output Distributor to skip to the EOF for the current file to be (or being) processed. It is not necessary for the Output Distributor to receive a HOLD directive prior to the CANCEL. All subsequent processing will then take place in the usual manner.

(d) RESTART

This primitive will direct the Output Distributor to reset the input pointer to the beginning of the file being (or to be - which then becomes a null operation) processed. It is not necessary for the Output Distributor to receive a HOLD directive prior to the RESTART.

(e) SINGLE (n)

This primitive is to be used with an output unit that is a printer. If the primitive is accepted by an Output Distributor which is being used for any other device, a null operation will result. In the case of a printer, this primitive will direct the Output Distributor to place a blank at the beginning of each record of output so as to effect a single spacing of the listing. The primitive will take effect immediately as soon as it is received. It is not necessary for the Output Distributor to receive a HOLD directive prior to the SINGLE. The SINGLE directive will lapse after the start of n files. If n is one and a RESTART directive is issued, the file will be printed in the normal manner. A new SINGLE directive will override any previous SINGLE directive. If a SINGLE directive with a value for n of zero (explicit - not default) is received, the rest of the file will be printed in the normal manner. The default value for n is one.

(f) ROLL (n)

This primitive will direct the Output Distributor to skip n records of output to the unit (n pages if the unit is a printer). The parameter n may be negative in which case the output will be repeated. If n is such that the extent of the file being output is exceeded (in either direction), the default will be to the start or end of the file. The default value for n is one.

(g) PROCESS (n)

This primitive will direct the Output Distributor to transfer the contents of the next n records of output to the unit (n pages if the unit is a printer or the file type is print) after which the Output Distributor will go into a HOLD condition. It is not necessary for the Output Distributor to receive a HOLD directive prior to the PROCESS. Note also that if the Output Distributor is in a HOLD condition, the PROCESS directive will not cause any data to be transferred. Only a CONTINUE directive has that effect. The default value for n is one.

(h) PROCEED (n)

This primitive will direct the Output Distributor to transfer the contents of the next n files to the unit after which the Output Distributor will go into a HOLD condition. It is not necessary for the Output Distributor to receive a HOLD directive prior to the PROCEED. Note also that if the Output Distributor is in a HOLD condition, the PROCEED directive will not cause any data to be transferred. Only a CONTINUE directive has that effect. The default value for n is one.

(i) FINISH

This primitive will direct the Output Distributor to terminate itself at the start of the next file if the Output Distributor was about to go into a HOLD condition. It is not necessary for the Output Distributor to receive a HOLD directive prior to the FINISH. Note that if the Output Distributor is in the middle of processing a file, a FINISH directive will not by itself immediately terminate processing. A PROCEED (1) followed by a RESTART must also be issued which would then preserve the file for future transfer. Alternatively, a PROCEED (1) followed by a CANCEL could be issued which would remove that entry from the output queue. If the Output Distributor is in a HOLD condition at the start of a new file (a RESTART or CANCEL directive will force it to the start of a new file), a FINISH directive will direct the Output Distributor to terminate itself immediately.

(j) FORM (type)

to be supplied

Conversion Routine Types

(a) COPY

COPY does not change the data format in any way. Its primary purpose is to allow for any file of any organization to be transferred to any unit (with presumably a default organization of sequential). In addition, when the output record exceeds the physical device size, the extra characters will be placed upon succeeding output records.

In the case of a printer, these succeeding records will be preceded by a blank character.

(b) COPYTK

COPYTK is identical to COPY, but with the additional feature that each record of data transmitted will be truncated to a specified number of characters (probably the physical device size). This feature would be used in the case of a punch unit being the output medium and the source file having more than the allowed number of characters per record.

(c) COPYSB

COPYSB is identical to COPYTK, but with the additional feature that each record of data transmitted will be preceded by a blank character.

(d) DUMP

DUMP cannot be defined at this point. It is expected to allow reformatting of the data in a file such that object files and libraries (to give but two examples) would have a meaningful listing if output was placed on a printer.

File Types

(a) Source

(b) Print

(c) Punch

(d) Terminal

(e) Object Program

(f) Library of Programs

Output Unit Types

(a) Printer

(b) Punch

(c) Terminal

Conversion Matrix

	printer	punch	terminal
null	COPYTK	COPYTK	COPYTK
source	COPYSB	COPYTK	COPYTK
print	COPY	COPYTK	COPYTK
punch	COPYSB	COPY	COPYTK
terminal	COPYSB	COPYTK	COPY
object	DUMP	DUMP	DUMP
library	DUMP	DUMP	DUMP

7.9 SYSTEM ACCESS MANAGER

System Access Manager runs as a single task within the System Job. Its primary purpose is the recognition of new input from system input devices and the submission of such inputs as new jobs to the system.

7.9.1 DESCRIPTION OF FUNCTIONS

The major function of System Access Manager is the recognition of events caused by the Active Device Detector indicating that a system input device is ready to transmit data to the system. Occurrence of such events will cause System Access Manager to scan the System Input Device list to determine which device became ready. The actions of System Access Manager will be different depending upon whether the input is from a batch or interactive source.

7.9.1.1 Interactive Input

If the input medium is an interactive terminal then System Access Manager will declare a uniquely named Job Control Block and a uniquely named File Control Block in System Global LNS. The File associated with this File Control Block is the interactive terminal and will be specified as the Standard Input File on the SUBMIT request. System Acess Manager will then issue the SUBMIT request to Job Management in order that the input will be processed.

A count will be maintained of the number of interactive devices logged in at any one time. If this count exceeds a specified system limit, then a message will be sent to the user requesting him to try again at a later time and the user's line will be disconnected.

7.9.1.2 Batch Input

If the input medium is a batch device, then System Access Manager will declare a uniquely named Job Control Block and a uniquely named File Control Block in System Global LNS and will then invoke the Input Stager to process the input from that device. Upon return from Input Stager, System Access Manager will issue a SUBMIT request to Job Management to process the file into which the Input Stager has placed a logical input stream. System Access Manager will then reinvoke the Input Stager in case there is more input on the same device.

A Count will be maintained of the number of activations of Input Stager. If the number of active Input Stagers reaches a system defined maximum then System Access Manager will queue any further requests for input until such time as there are resources to process the input.

7.9.2 EXTERNAL DATA STRUCTURES

The following externally defined data structures are used by System Access Manager.

Job Control Block.

System Access Manager declares a Job Control Block for each new job entering the system. The following information is placed in the Job Control Block.

- o Job Class, that is Batch or Interactive.
- o PII, the Primal Invocator Identity.

File Control Block.

System Access Manager declares a File Control Block for each Input Device and also a File Control Block for every mass storage file required by Input Stager. The individual fields within each File Control Block are filled in by Data Management requests. File Formats are defined below.

Event Control Block.

Event Control Blocks will be used to communicate between System Access Manager and the Active Device Detector.

Subtask Control Block.

System Access Manager will have a Subtask Control Block for use in the SPAWN request to invoke Input Stager.

Configuration Manager Unit Tables.

System Access Manager will access these Unit Tables in order to determine which System Input Device has become ready for input.

7.9.3 INTERNAL DATA STRUCTURES

System Access Manager will have one data structure internal to itself. This data structure will be a table of pointers to the System Input Device Tables, each pointer pointing at a system input device which has been placed 'Onsystem'. The information required by System Access Manager in these tables is specified below.

System Input Device Tables.

System Access Manager will require information specific to the devices from which it receives input. A preliminary idea of the type of information required is given below. These descriptions are by no means rigorous or definitive, but give some notion of the information required.

```
Type.
     Device_Classes = ( Interactive_Device,Batch_Device ) ,
     Device_Types = ( Card,Tape,Disk,Terminal,etc ) ,

     Primal_Invocator_Identity = Record
          Site : Integer ,
          Terminal : Integer ,
          Unit : Integer ,
     Recend ,


     System_Input_Device_Description = Record
          Device_Identifier : Primal_Invocator_Identity ,
          Ready_Indicator : Boolean ,
          Device_Type : Device_Types ,
          Device_Class : Device_Classes ,
     Recend ;
```

7.9.4 OPERATOR COMMUNICATIONS INTERFACE

System Access Manager has three request processors which interface with Operator Communications. These are :

'Onsystem' request.

'Offsystem' request.

'Offline' request.

The 'Onsystem' request will notify System Access Manager that a device is a system input device and will cause System Access Manager to make an entry in its system input device table. The 'Offsystem' request will cause the relevant table entry to be deleted. The 'Offline' request is necessary because it is feasible that a device which is 'Onsystem' can be taken 'Offline' without first taking it 'Offsystem'.

7.9.4.1 'Onsystem' Request Processor

The format of the 'Onsystem' request is as follows.

SA#ONSYSTEM(Unit_Descriptor , Status)

Unit_Descriptor : The Unit_Descriptor parameter the LNS descriptor of the device in Configuration Manager's Unit tables.

Status : Status is the identifier of the status record returned to the caller by System Access Manager. The status codes are defined below.

7.9.4.2 'Offsystem' Request Processor

The format of the 'Offsystem' request is as follows.

SA#OFFSYSTEM(Unit_Descriptor , Status)

Unit_Descriptor : The Unit_Descriptor parameter the LNS descriptor of the device in Configuration Manager's Unit tables.

Status : Status is the identifier of the status record returned to the caller by System Access Manager. The status codes are defined below.

7.9.4.3 'Offline' Request Processor

The format of the 'Offline' request is as follows.

SA#OFFLINE(Unit_Descriptor , Status)

Unit_Descriptor : The Unit_Descriptor parameter the LNS descriptor of the device in Configuration Manager's Unit tables.

Status : Status is the identifier of the status record returned to the caller by System Access Manager. The status codes are defined below.

7.9.4.3.1 STATUS CODES.

The Status Codes returned by System Access Manager are as follows.

0 SA 000	Request successfully executed.
4 SA 001	Onsystem request for device already Onsystem.
4 SA 002	Offsystem request for device already Offsystem.
4 SA 003	Offline request for device already Offsystem.
8 SA 004	Onsystem request for device which is not an Input Device.
8 SA 005	Offsystem request fo~ a device which is not an Input Device.

7.9.5 SUBMIT FORMATS.

There are two separate formats for the SUBMIT request to Job Management depending on whether the input is from a batch or an interactive device. The two different formats are as follows.

Batch Input.

JM#SUBMIT(FCB , JCB , ENQ=Q , DISP=P , Status)

This request indicates that the file specified by FCB will be purged on job completion and that the job will be queued if there are insufficient resources to process the job.

Interactive Input.

JM#SUBMIT(FCB , JCB , ENQ=I , DISP=K , Status)

This request indicates that in the event of there being insufficient resources to process the job,submit will be rejected.

7.9.5.1 Input Stager.

Input Stager runs as a subtask of System Access Manager. It is responsible for transferring data from system input devices to mass storage staging files.

The functions performed by Input Stager are as follows.

- o Creates a file whose File Control Block has been declared by System Access Manager.
- o Copies data from the Input File to the mass storage staging File until a 'fence' card has been read.
- o Saves the mass storage staging File.
- o Returns to System Access Manager with an indicator stating whether 'device end' was encountered on the input device.
- o Input Stager will report on staged jobs in the system log.

7.9.5.2 Hedge Cards.

To cater for those cases where large quantities of data are being staged from cards and the user wishes to take precautions against system crashes, a new type of data separator will be provided. This new type of card is known as a HEDGE card. Input Stager will always remember the input deck as far as the last Hedge card which it encounters, such that if a crash does occur, the user need only reinput the portion of his data following the last successfully read Hedge card.

In keeping with 'Fence' cards, Hedge cards are identified uniquely by some combination of punching which requires multipunching. A Hedge card also requires two additional items of information, these two extra items being User's Logical Deck Name and Hedge Card Name.

User's Logical Deck Name is required since the system stages input to uniquely named files. It would be difficult in some cases to inform the user of the name of the file onto which his input deck has been staged, so the user is allowed to provide a Logical Name for his particular deck. Input Stager will keep a record of Logical Deck Names associated with each input device such that the user may access the name of the staging file associated with his Logical Deck.

The first Hedge Card encountered in a deck will be the one which identifies the Logical Deck Name. Each subsequent Hedge Card can optionally omit the Logical Deck Name, the one first quoted being used. Each Hedge Card encountered in the input deck must have a unique name. In the event of a system crash the user may then quote that Hedge Card Name as the position at which he wishes to continue staging his input. The exact format of Hedge cards will be defined at a later stage.

7.9.5.3 File Formats.

Two types of File are created by System Access Manager. One is the file from System Input Devices and the other is the mass storage files written by Input Stager.

System Input Device Files.

The general characteristics of this type of file will be.

- File Control Block declared with a unique name in System Global LNS.
- File Organization will be Sequential.
- Usage will be Shared Read.
- Access Method will be Sequential at the Record level.

Mass Storage Staging File.

The general characteristics of this type of file will be.

- File Control Block declared with a unique name in System Global LNS.
- File will reside on System Volume Set.
- File Organization to be Sequential.
- Access Method will be Sequential at the Record level.
- Usage will be Exclusive Write.
- Error Processing will be left to System error handling procedures.

8.0 JOB MANAGEMENT STRUCTURES

This section identifies and specifies the structures which the facilities of Job Management utilize in the performance of their functions.

8.1 JOB CONTROL BLOCK (JCB)

8.1.1 STRUCTURE TYPE

A system defined LNS record.

8.1.2 UNIT OF ASSIGNMENT

One JCB for each job known to the system.

8.1.3 LOCATION OF STRUCTURE

System Global LNS

8.1.4 RESIDENCY CHARACTERISTICS

Pageable; not swapped with the job.

8.1.5 STRUCTURE ASSIGNMENT

A JCB is declared by the SUBMIT Request Processor for jobs submitted from within user jobs. For the system's submission of staged batch jobs and interactive jobs, a JCB is declared by the STAGER program and the SYSTEM ACCESS MANAGER, respectively.

8.1.6 LIFECYCLE

Once declared, the JCB exists until the job with which it is associated has terminated <u>and</u> until all dependency threads originating in it, or continuing through it, are no longer required by the system.

8.1.7 CONTENTS

- o External name of the job; this is the unique name assigned by the system to the JCB itself.
- o Job status
  - known but not established
  - established but not initiated
  - initiated
  - completed
- o Times
  - first known to the system
  - established
  - initiated
  - completed
- o Initial priority
- o User identifier
- o Account identifier
- o Queue-for-establishment option designator
- o Primary Input File disposition option designator
- o Logical Sense Switches
- o Job class (batch/interactive)
- o Job type code (standard user Job, Subsystem Supervisor Job, System Job, Diagnostic Job)
- o Primal Invocator Identifier (PII)
- o Cataloged named of the job's Primary Input File
- o Identifier of FCB of the job's Pool File
- o Identifier of Job's Swap Segment
- o Known Job List (KJL) ordinal of KJL entry assoc. with this job
- o Pointer to Job Gate Table in the job's address space
- o Pointer to Job Stack Table in job's address space
- o Origin of the job's Established Program Control Block (EPCB) thread
- o Job Resource Limit Record
  - Processor time limit
  - Memory limit
  - Peripheral Limit Control Elements (one per peripheral type)
    - -- specified limit

-- number shared
-- number assigned
-- number claimed

- o Address of the JCB of the Job which submitted this Job; for root Jobs, this value is zero
- o Address of a JCB in a forward thread of JCBs which correspond to other Jobs submitted by the submittor of this Job
- o Address of a JCB in a backward thread of JCBs which correspond to other Jobs submitted by the submittor of this Job
- o Address of a JCB which constitutes the origin of a thread of JCBs which correspond to Jobs submitted by this Job

8.2 KNOWN JOB LIST (KJL)

8.2.1 STRUCTURE TYPE

KJL is comprised of a fixed number of fixed length entries. Every entry in KJL is a member of one of five threaded lists: (1) those entries which are available for assignment, (2) those entries associated with queued jobs, (3) those entries associated with deferred jobs, (4) those entries associated with running jobs, and (5) those entries which are in a state of transition between the other four threaded lists.

8.2.2 UNIT OF ASSIGNMENT

One entry for every Job known to the system. Entries having ordinals 1 and 2 are permanently assigned to System Monitor and the System Job, respectively.

8.2.3 LOCATION OF STRUCTURE

To be determined *************

8.2.4 RESIDENCY CHARACTERISTICS

Pageable; not swapped with the associated Job

8.2.5 STRUCTURE ASSIGNMENT

A KJL entry is assigned for each Job either at the time the Job is established or at the time it is queued for delayed establishment. KJL entries are acquired for a job by the Job Establisher task of the System Job.

8.2.6 LIFECYCLE

The KJL entry for a Job exists until the Job is collapsed by the Job Collapser task of the System Job.

8.2.7 CONTENTS

- o Identifier of the JCB of the Job with which the KJL entry is associated.
- o Job Status
  - - queued for establishment
  - - established
    - .. swapped out
    - .. swapped in
      - -- active
      - -- inactive
- o Pointer to previous KJL entry in same thread as this entry
- o Pointer to next KJL entry in same thread as this entry
- o KJL thread identifier
- o Swap File identifier
- o Working Set Size
- o Major Time Slice
- o Remaining amount of major time slice
- o Minor Time Slice
- o Time selection
- o Time thread
- o Job identifier (<KJL ordinal> concatenated with <sequence number>)
- o Current Job priority
- o Entry Lock Indicator
- o Identifier of element holding a lock
- o Base Job priority
- o Amount of central processor time used
- o Amount of memory-time used

8.3 JOB STATE THREAD ORIGIN TABLE (JSTOT)

8.3.1 STRUCTURE TYPE

JSTOT is comprised of a fixed number of fixed length entries.

8.3.2 UNIT OF ASSIGNMENT

One JSTOT in the system

8.3.3 LOCATION OF STRUCTURES

Same as KJL

8.3.4 RESIDENCY CHARACTERISTICS

Pageable

8.3.5 STRUCTURE ASSIGNMENT

JSTOT is constructed during system loading operation; it is permanently assigned to the System Job.

8.3.6 LIFECYCLE

Exists throughout the life of the system.

8.3.7 CONTENTS

o PVA of the origin of the Known Job List (KJL)
o Length of a KJL entry
o Number of entries in KJL

o Address of highest priority entry in KJL "available" thread
o Address of highest priority entry in KJL "neutral" thread
o Address of highest priority entry in KJL "queued" thread
o Address of highest priority entry in KJL "running" thread
o Address of highest priority entry in KJL "deferred" thread

8.4 RUNNING JOB ORDINAL TABLE (RJOT)

8.4.1 STRUCTURE TYPE

RJOT is comprised of a fixed number of fixed length entries.

8.4.2 UNIT OF ASSIGNMENT

One entry for each job running in the system; one RJOT entry is permanently dedicated to System Monitor.

8.4.3 LOCATION OF STRUCTURE

Segment 5 of System Monitor address space.

8.4.4 RESIDENCY CHARACTERISTICS

Not pageable; swapped with the associated job.

8.4.5 STRUCTURE ASSIGNMENT

An RJOT entry is assigned to a Job by System Monitor as a result of Job Establisher issuing a CREATE_ADDRESS_SPACE request during the Job establishment process.

8.4.6 LIFECYCLE

An RJOT entry remains assigned to a Job during the time which the Job is running. The entry is released after its contents are transferred to the Job's swap segment during the process of swapping-out the Job. An RJOT entry is assigned during the process of swapping-in a Job and the previous entry contents are obtained from the Job's swap segment.

8.4.7 CONTENTS

Field Name   Description

Job_ID   Job's Type, corresponds to Job Descriptor's entry number within KJL.

Type   Job Identifier. Currently the following job types are defined

- o Standard User Job
- o Subsystem Supervisor
- o System Job
- o Diagnostic Job
- o Deadstart/Recovery Job

Status   Job's Status_Indicators, these are the following:

- o RJOT Interlocked
- o Active Job
- o Going Inactive/Active
- o Full Swap-Out in progress
- o Full Swap-In in progress
- o No time limit
- o No swap permitted
- o No time slice
- o One-Control Point-a-time dispatching
- o No forced page-steal
- o Page_Interrupt_RP in action
- o Save_Signals

SDT_Ordinal   Relative pointer to Segment Descriptor Table of this job within Segment 3 of System Monitor.

TSL   Table_Size_Limit, sets maximum allowable size of job owned tables within system space (modulo 256 bytes).

CTL   Current_Table_Size

PATT   Page_Age_Tick_Time, determines the aging_rate of pages belonging to a job's Working_Set. (Set by Job Scheduler)

PICI   Page_Interrupt_Count_Increment. Number of page interrupts accumulated since last examination by Page Control.

WSSL   Working_Set_Size_Limit, if Working Set Size exceeds this limit, Job Scheduler is notified (in page size).

CWSS   Current_Working_Set_Size

JTL   Job_Time_Limit. Dispatcher compares the sum of Task_Execution_Times against JTL on expiration of a Task's time-slice, Job Scheduler is notified when JTL is exceeded.

LPIT   Last_Page_Interrupt_Time, sum of Task Execution Times is recorded here on a Page Interrupt.

PIAS   Page_Interrupt_Accumulation_Start time in Task Execution Times.

WSL   Working_Set_Link, relative pointer into Memory_Map. Used by Page Control to find the head of WS page chain.

SSID   Swap Segment's unique identifier is kept here while a job is being deferred or made running.

or

SSO   Swap_Segment_Ordinal, relative Pointer into ASNT to find Swap Segment of a running job.