# Job Monitoring MIB

## (This cover page is *not* part of the Internet-Draft)

From:   Tom Hastings

Date:   0604/0924/97

Version: 0.821

File:   ftp://ftp.pwg.org/pub/jmp/mibs/jmp-mib.doc  .pdf    jmp-mibr.doc  .pdf .pdr

Status:   FifthFourth draft MIB that corresponds to the changes agreed to at the JMP meeting, on Friday, 5/16/97.  at the JMP meeting, 04/04/97 in Austin.The major changes were to eliminated duplicates between the Job State table and the Attribute table, and to move all mandatory integer attributes to the Job Table, leaving only the **jobOwner** string attribute as MANDATORY in the Attribute table.  The Job State table has been renamed back to the Job table, since it has more than just state now. Harry Lewis's changes to eliminate the Queue and Completed tables and to replace the Job table with the Job ID and Job State table have been incorporated.  See the change history in the separate file: changes.doc  .pdf.  The Internet-Draft was not posted in time and with these changes, we did not present any MIB document at the IETF meeting on 04/08/97 in Memphis.  Instead we presented slides on the current status explaining the tables, which are:  General, Job ID, Job State, and Attributes.

I've also produced a variation on this document which has all variable font (**jmp-mibv.doc .pdf**) without revision marks. This is the version that the JMP should use to make comments.  It has line numbers.

The MIB has been greatly simplified so that now there are only 1713 objects in the MIB.  There are 7178 attributes: , of which only 1 is8 are MANDATORY and 70 are OPTIONAL.

I've removed the issues from the document and placed them in a separate document: issues.doc  .pdf.  There are very few issues remaining.  I've added a few issues from the e-mail since the last meeting.

The actual specifications of each object needs line-by-line review.  We did *not* have time for such review at the 11/08/96 or the 01/08/97 meeting as indicated in the minutes.  The group wanted to wait until this specification is re-formatted into a MIB.

I've moved the full ISO DPA specifications to a separate document. I've also copied map-summ.doc into another document so we can compare the Job Monitoring objects with the job submission protocols and keep the object names updated in that summary.

We moved more objects into the Resource Table, now called the Attribute Table, since more than resources are in it.  I've not used revision marks for such moves, but only for changes within each description of what had been an object and what now is an enum.

38  INTERNET-DRAFT                                          Ron Bergman
39                                                    Dataproducts Corp.
40                                                        Tom Hastings
41                                                   Xerox Corporation
42                                                       Scott Isaacson
43                                                         Novell, Inc.
44                                                         Harry Lewis
45                                                           IBM Corp.
46                                                          April 1997
47

48               **Job Monitoring MIB - V0.821**

49          **<draft-ietf-printmib-job-monitor-010.txt>**

50              **Expires DecOct 1024, 1997**

51

52  **Status of this Memo**

53       This document is an Internet-Draft.  Internet-Drafts are working documents of the
54       Internet Engineering Task Force (IETF), its areas, and its working groups.  Note
55       that other groups may also distribute working documents as Internet-Drafts.

56       Internet-Drafts are draft documents valid for a maximum of six months and may be
57       updated, replaced, or obsoleted by other documents at any time.  It is
58       inappropriate to use Internet-Drafts as reference material or to cite them other than
59       as "work in progress."

60       To learn the current status of any Internet-Draft, please check the "1id-
61       abstracts.txt" listing contained in the Internet-Drafts Shadow Directories on
62       ftp.is.co.za (Africa), nic.nordu.net (Europe), munnari.oz.au (Pacific Rim),
63       ds.internic.net (US East Coast), or ftp.isi.edu (US West Coast).

64                              **Abstract**

65       This Internet-Draft specifies a set of 1713 SNMP MIB objects for (1) monitoring
66       the status and progress of print jobs (2) obtaining resource requirements before a
67       job is processed, (3) monitoring resource consumption while a job is being
68       processed and (4) collecting resource accounting data after the completion of a
69       job.  This MIB is intended to be implemented (1) in a printer or (2) in a server that
70       supports one or more printers.  Use of the object set is not limited to printing.
71       However, support for services other than printing is outside the scope of this Job
72       Monitoring MIB.  Future extensions to this MIB may include, but are not limited
73       to, fax machines and scanners.

74

## TABLE OF CONTENTS

232

233                              **Job Monitoring MIB**

234    ## 1. Introduction

235    The Job Monitoring MIB consists of a 65-object General Group, a 2-object Job
236    Submission ID Group, a 74-object Job ~~State~~ Group, and a 2-object Attribute Group.
237    Each group is a table.  The General Group contains general information that applies to all
238    jobs in a job set.  The Job Submission ID table maps the job submission ID that the client
239    uses to identify a job to the jmJobIndex that the Job Monitoring Agent uses to identify
240    jobs in the Job ~~State~~ and Attribute tables.  The Job ~~State~~ table contains the mandatory
241    integer job state and status objects~~copies of three salient attributes for each job's current~~
242    ~~state~~.  The Attribute table consists of multiple entries per job that specify (1) job and
243    document identification and parameters,  (2) requested resources, and (3) consumed
244    resources during and after job processing/printing.  One MANDATORY attribute and 70
245    OPTIONAL attributes are defined as textual conventions.

246    The Job Monitoring MIB is intended to be instrumented by an agent within a printer or the
247    first server closest to the printer, where the printer is either directly connected to the
248    server only or the printer does not contain the job monitoring MIB agent.  It is
249    recommended that implementations place the SNMP agent as close as possible to the
250    processing of the print job.  This MIB applies to printers with and without spooling
251    capabilities.  This MIB is designed to be compatible with most current commonly-used job
252    submission protocols.  In most environments that support high function job submission/job
253    control protocols, like ISO DPA[2], those protocols would be used to monitor and
254    manage print jobs rather than using the Job Monitoring MIB.

255    ### 1.1  Types of Information in the MIB

256    The job MIB is intended to provide the following information for the indicated Role
257    Models in the Printer MIB[1] (~~Refer to RFC 1759,~~ Appendix D - Roles of Users).

258        User:

259            Provide the ability to identify the least busy printer.  The user will be able to
260            determine the number and size of jobs waiting for each printer.  No attempt is
261            made to actually predict the length of time that jobs will take.

262            Provide the ability to identify the current status of the user's job (user queries).

263            Provide a timely indication~~notification~~ that the job has completed and where it
264            can be found.

265            Provide error and diagnostic information for jobs that did not successfully
266            complete.

267      Operator:

268            Provide a presentation of the state of all the jobs in the print system.

269            Provide the ability to identify the user that submitted the print job.

270            Provide the ability to identify the resources required by each job.

271            Provide the ability to define which physical printers are candidates for the print
272            job.

273            Provide some idea of how long each job will take.  However, exact estimates of
274            time to process a job is not being attempted.  Instead, objects are included that
275            allow the operator to be able to make gross estimates.

276      Capacity Planner:

277            Provide the ability to determine printer utilization as a function of time.

278            Provide the ability to determine how long jobs wait before starting to print.

279      Accountant:

280            Provide information to allow the creation of a record of resources consumed and
281            printer usage data for charging users or groups for resources consumed.

282            Provide information to allow the prediction of consumable usage and resource
283            need.

284   The MIB supports printers that can contain more than one job at a time, but still be usable
285   for low end printers that only contain a single job at a time.  In particular, the MIB
286   supports the needs of Windows and other PC environments for managing low-end
287   networked devices without unnecessary overhead or complexity, while also providing for
288   higher end systems and devices.


289   **1.2  Types of Job Monitoring Applications**

290   The Job Monitoring MIB is designed for the following types of monitoring applications:

291      1.  monitor a single job starting when the job is submitted and finishing a defined
292            period after the job completes.  The Job Submission ID table provides the map to
293            find the specific job to be monitored.

294      2.  monitor all 'active' of the jobs in a queue, which this specification is generalizesd to
295            a "job set".  End users may use such a program when selecting a least busy printer,
296            so the MIB is designed for such a program to start up quickly and find the
297            information needed quickly without having to read all (completed) jobs in order to
298            find the active jobs.  System operators may also use such a program, in which case
299            it would be running for a long period of time and may also be interested in the jobs

300      that have completed.  Finally such a program may be co-located with the printer to
301      provide an enhanced console and logging capability.

302   3.   collect resource usage for accounting or system utilization purposes that copy the
303        completed job statistics to an accounting system. It is recognized that depending on
304        accounting programs to copy MIB data during the job-retention period is
305        somewhat unreliable, since the accounting program may not be running (or may
306        have crashed).  Such a program is expected to keep a shadow copy of the entire
307        Job **Attribute** table including ~~canceled and~~ **completed, canceled, and aborted**
308        jobs which the program updates on each polling cycle.  Such a program polls at the
309        rate of the persistence of the **Attribute** table.  The design is not optimized to help
310        such an application determine which jobs are **completed,** ~~or~~ **canceled, or aborted**.
311        Instead, the application SHALL query each job that the application's shadow copy
312        shows was not **complete,** ~~or~~ **canceled, or aborted** at the previous poll cycle to see
313        if it is now **complete** or **canceled**, plus any new jobs that have been submitted.

314   The MIB provides a set of objects that represent a compatible subset of job and document
315   attributes of the ISO DPA standard[2] and the Internet Printing Protocol (IPP)[3], so that
316   coherence is maintained between these two protocols and the information presented to end
317   users and system operators by monitoring applications.  However, the job monitoring MIB
318   is intended to be used with printers that implement other job submitting and management
319   protocols, such as IEEE 1284.1 (TIPSI)[4], as well as with ones that do implement ISO
320   DPA.  So nothing in the job monitoring MIB ~~SHALL~~ requires implementation of the ISO
321   DPA or IPP protocols.

322   The MIB is designed so that an additional MIB(s) can be specified in the future for
323   monitoring multi-function (scan, FAX, copy) jobs as an augmentation to this MIB.

## 2.  Terminology and Job Model

325   This section defines the terms that are used in this specification and the general model for
326   jobs.

327      NOTE - Existing systems use conflicting terms, so these terms are drawn from the ISO
328      10175 Document Printing Application (DPA) standard[2].  For example, PostScript
329      systems use the term *session* for what we call a *job* in this specification and the term
330      *job* to mean what we call a *document* in this ~~specification~~paper.  PJL systems use the
331      term *job to mean what we call a job in this specification.  PJL also supports multiple*
332      *documents per job, but does not support specifying per-document attributes*
333      *independently for each document*.

334   A *job* is a unit of work whose results are expected together without interjection of
335   unrelated results.  A *client* is able to specify *job instructions* that apply to the job as a

336   whole.  Proscriptive instructions specify how, when, and where the job is to be printed.
337   Descriptive instructions describe the job.  A job contains one or more *documents*.

338   A *job set* is a set of jobs that are queued and scheduled together according to a specified
339   scheduling algorithm for a specified device or set of devices.  For implementations that
340   embed the SNMP agent in the device, the MIB job set normally represents *all* the jobs
341   known to the device, so that the implementation only implements a single job set which
342   MAY be identified with a hard-coded value **1**.  If the SNMP agent is implemented in a
343   server that controls one or more devices, each MIB job set represents a job queue for (1)
344   a specific device or (2) set of devices, if the server uses a single queue to load balance
345   between several devices.  Each job set is disjoint; no job SHALL be represented in more
346   than one MIB job set.

347   A *document* is a sub-section within a job.  A document contains print data and *document*
348   *instructions* that apply to just the document.  The *client* is able to specify document
349   instructions separately for each document in a job.  Proscriptive instructions specify how
350   the document is to be processed and printed by the *server*.  Descriptive instructions
351   describe the document.  Server implementation of more than one document per job is
352   optional.

353   A *client* is the network entity that *end users* use to submit jobs to *spoolers*, *servers*, or
354   *printers* and other *devices*, depending on the configuration, using any job submission
355   protocol.

356   A *server* is a network entity that accepts jobs from clients and in turn submits the jobs to
357   *printers* and other *devices*.  A server MAY be a printer *supervisor* control program, or a
358   print *spooler*.

359   A *device* is a hardware entity that (1) interfaces to humans in human perceptible means,
360   such as produces marks on paper, scans marks on paper to produce an electronic
361   representations, or writes CD-ROMs or (2) interfaces to a network, such as sends FAX
362   data to another FAX device.

363   A *printer* is a *device* that puts marks on media.

364   A *supervisor* is a server that contains a control program that controls a printer or other
365   device.  A supervisor is a client to the printer or other device.

366   A *spooler* is a server that accepts jobs, spools the data, and decides when and on which
367   printer to print the job.  A spooler is a client to a printer or a printer supervisor, depending
368   on implementation.

369   *Spooling* is the act of a *device* or *server* of (1) accepting jobs and (2) writing the job's
370   attributes and document data on to secondary storage.

371   *Queuing* is the act of a *device* or *server* of ordering (queuing) the jobs for the purposes of
372   scheduling the jobs to be processed.

373   A *monitor* or *job monitoring application*  is the network entity that End Users, System
374   Operators, Accountants, Asset Managers, and Capacity Planners use to monitor jobs using
375   SNMP.  A monitor MAY be either a separate application or MAY be part of the client
376   that also submits jobs.

377   An *agent* is the network entity that accepts SNMP requests from a *monitor* and
378   implements the Job Monitoring MIB by instrumenting a *server* or a *device*.

379   A *proxy* is an agent that acts as a concentrator for one or more other agents by accepting
380   SNMP operations on the behalf of one or more other agents, forwarding them on to those
381   other agents, gathering responses from those other agents and returning them to the
382   original requesting monitor.

383   A *user* is a person that uses a client or a monitor.

384   An *end user* is a user that uses a client to submit a print job.

385   A *system operator* is a user that uses a monitor to monitor the system and carries out tasks
386   to keep the system running.

387   A *system administr*ator is a user that specifies policy for the system.

388   A *job instruction* is an instruction specifying how, when, or where the job is to be
389   processed.  Job instructions MAY be passed in the job submission protocol or MAY be
390   embedded in the document data or a combination depending on the job submission
391   protocol and implementation.

392   A *document instruction* is an instruction specifying how to process the document.
393   Document instructions MAY be passed in the job submission protocol separate from the
394   actual document data, or MAY be embedded in the document data or a combination,
395   depending on the job submission protocol and implementation.

396   An *SNMP information object* is a name, value-pair that specifies an action, a status, or a
397   condition in an SNMP MIB.  Objects are identified in SNMP by an OBJECT
398   IDENTIFIER.

399   An *attribute* is a name, value-pair that specifies an instruction, a status, or a condition of a
400   job or a document that has been submitted to a server or device.  A particular attribute
401   NEED NOT be present in each job instance.  In other words, attributes are present in a
402   job instance only when there is a need to express the value, either because (1) the client
403   supplied a value in the job submission protocol, (2) the document data contained an
404   embedded attribute, or (3) the server or device supplied a default value.  An agent SHALL
405   represent an attribute as an entry (row) in the Attribute table in this MIB in which entries
406   are present only when necessary.  Attributes are identified in this MIB by an enum..

407   *Job monitoring* using SNMP is (1) identifying jobs within the serial streams of data being
408   processed by the server, printer or other devices, (2) creating "rows" in the job table for

409  each job, and (3) recording information, known by the agent, about the processing of the
410  job in that "row".

411  *Job accounting* is recording what happens to the job during the processing and printing of
412  the job.

### 3.  System Configurations for the Job Monitoring MIB

414  This section enumerates the three configurations infor which the Job Monitoring MIB is
415  intended to be used.  To simplify the pictures, the *devices* are shown as *printers*.  See
416  Goals section.

417  The diagram in the Printer MIB[1] entitled: "One Printer's View of the Network" is
418  assumed for this MIB as well.  Please refer to that diagram to aid in understanding the
419  following system configurations.

### 3.1  Configuration 1 - client-printer

421  In the **client-printer** configuration, the **client**(s) submit jobs directly to the printer, either
422  by some direct connect, or by network connection.  The **client-printer** configuration can
423  accommodate multiple job submitting **clients** in either of two ways:

424      1.  if each **client** relinquishes control of the Print Job Delivery Channel after each
425          job (or after a number of jobs)

426      2.  if the printer supports more than one Print Job Delivery Channel

427  The job submitting **client** and/or **monitoring application** monitor jobs by communicating
428  directly with an agent that is part of the printer.  The agent in the printer SHALL keep the
429  job in the Job Monitoring MIB as long as the job is in the Printer, and longer in order to
430  implement the **completed** state in which monitoring programs can copy out the
431  accounting data from the Job Monitoring MIB.

432
```
433              all           end-user      ######## SNMP query
434          +-------+       +--------+      ---- job submission
435          |monitor|       | client |
436          +---#---+       +--#--+--+
437              #              #   |
438              # ###########      |
439              # #               |
440      +==+===#=#=+==+           |
441      |  | agent |  |           |
442      |  +-------+  |           |
443      |   PRINTER   <--------+
444      |              | Print Job Delivery Channel
```

445                    |                    |
446                    +=============+
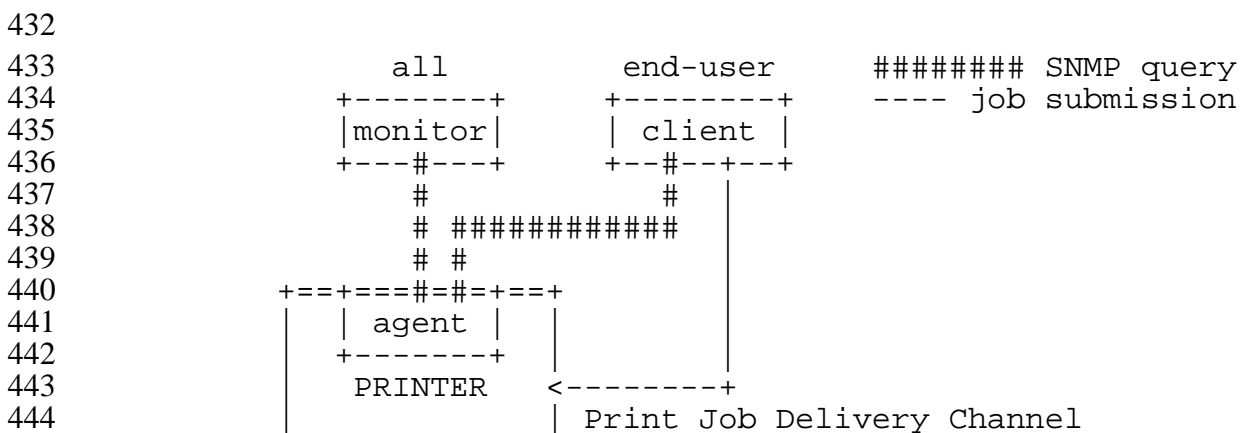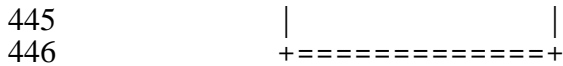
**Figure 3-1 - Configuration 1 - client-printer - agent in the printer**

448  The Job Monitoring MIB is designed to support the following relationships (not shown in
449  Figure 3-1):

450      1.  Multiple **clients** MAY submit jobs to a **printer**.
451      2.  Multiple **clients** MAY monitor a **printer**.
452      3.  Multiple **monitors** MAY monitor a **printer**.
453      4.  A **client** MAY submit jobs to multiple **printers**.
454      5.  A **monitor** MAY monitor multiple **printers**.


**3.2  Configuration 2 - client-server-printer - agent in the server**

456  In the **client-server-printer** configuration 2, the **client**(s) submit jobs to an intermediate
457  **server** by some network connection, *not* directly to the **printer**.  While configuration 2 is
458  included, the design center for this MIB is configurations 1 and 3,

459  The job submitting **client** and/or **monitoring application** monitor job by communicating
460  directly with:

461          1.  a Job Monitoring MIB agent that is part of the **server** (or a front for the
462              server)

463  There is no SNMP Job Monitoring MIB agent in the printer in configuration 2, at least
464  that the client or monitor are aware.  In this configuration, the agent SHALL return the
465  current values of the objects in the Job Monitoring MIB both for jobs the server keeps and
466  jobs that the server has submitted to the printer.  In configuration 2, the server keeps a
467  copy of the job during the time that the server has submitted the job to the printer.  Only
468  some time *after* the printer completes the job, SHALL the server remove the
469  representation of the job from the Job Monitoring MIB in the server.  The agent NEED
470  NOT access the printer, except when a monitor queries the agent using an SNMP Get for
471  an object in the Job Monitoring MIB.  Or the agent can subscribe to the notification events
472  that the printer generates and keep the Job Monitoring MIB update to date.  The agent in
473  the server SHALL keep the job in the Job Monitoring MIB as long as the job is in the
474  Printer, and longer in order to implement the **completed** state in which monitoring
475  programs can copy out the accounting data from the Job Monitoring MIB.
476
```
477          all              end-user
478       +-------+        +----------+
479       |monitor|        |  client  |      ######## SNMP query
480       +---+---#        +---#----+-+      **** non-SNMP cntrl
481           #              #      |        ---- job submission
482             #              #      |
```

```
483                          #        #     |
484                           #=====#=+==v==+
485                           | agent |     |
486                           +-------+      |
487                           |     server   |
488                           +----+-----+--+
489                    control *        |
490                    *********        |
491                        *            |
492            +========v====+          |
493            |             |          |
494            |             |          |
495            |   PRINTER   <---------+
496            |             | Print Job Delivery Channel
497            |             |
498            +=============+
```
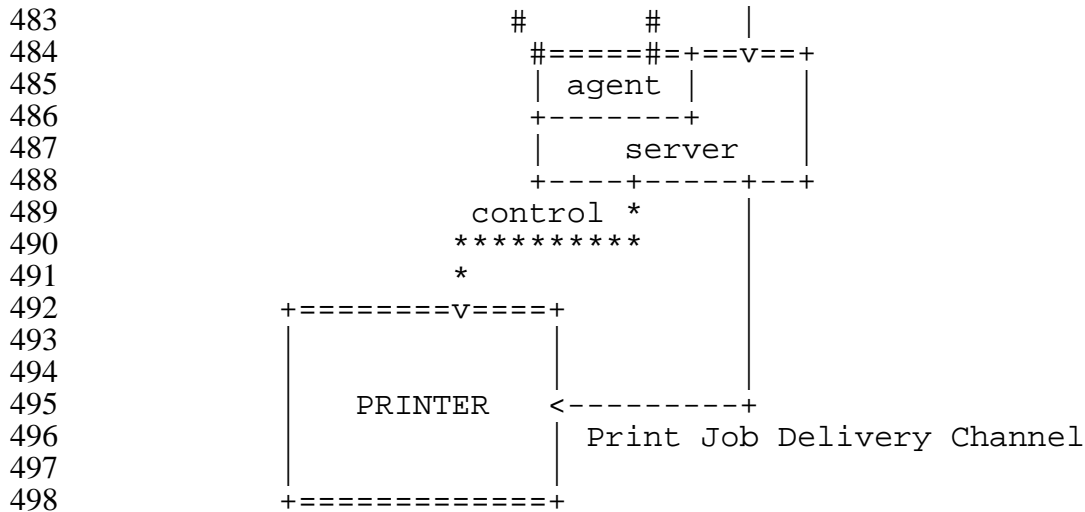
**Figure 3-2 - Configuration 2 - client-server-printer - agent in the server**

The Job Monitoring MIB is designed to support the following relationships (not shown in Figure 3-2):

1. Multiple **clients** MAY submit jobs to a **server**.
2. Multiple **clients** MAY monitor a **server**.
3. Multiple **monitors** MAY monitor a **server**.
4. A **client** MAY submit jobs to multiple **servers**.
5. A **monitor** MAY monitor multiple **servers**.
6. Multiple **servers** MAY submit jobs to a **printer**.
7. Multiple **servers** MAY control a **printer**.

**3.3  Configuration 3 - client-server-printer - client monitors printer agent and server**

In the **client-server-printer** configuration 3, the **client**(s) submit jobs to an intermediate **server** by some network connection, *not* directly to the **printer**.

The job submitting **client** and/or **monitoring application** monitor jobs by communicating directly with:

1. the server using some protocol to monitor jobs in the server that does not contain the Job Monitoring MIB AND

2. a Job Monitoring MIB agent that is part of the **printer** to monitor jobs after the server passes the jobs to the printer.  In such configurations, the server deletes its copy of the job from the server after submitting the job to the printer usually almost immediately (before the job does much processing, if any)**.**

There is no SNMP Job Monitoring MIB agent in the server in configuration 3, at least that the client or monitor are aware.  In this configuration, the agent (in the printer) SHALL

522   keep the values of the objects in the Job Monitoring MIB that the agent implements
523   updated for a job that the server has submitted to the printer.  The agent SHALL obtain
524   information about the jobs submitted to the printer from the server (either in the job
525   submission protocol, in the document data, or by direct query of the server), in order to
526   populate some of the objects the Job Monitoring MIB in the printer.  The agent in the
527   printer SHALL keep the job in the Job Monitoring MIB as long as the job is in the Printer,
528   and longer in order to implement the **completed** state in which monitoring programs can
529   copy out the accounting data from the Job Monitoring MIB.
530
```
531            all              end-user
532         +-------+        +----------+
533         |monitor|        |  client  |      ######## SNMP query
534         +---+---*        +---*----+-+      **** non-SNMP query
535             #      *          *    |       ---- job submission
536             #       *         *    |
537             #        *        *    |
538             #         *=====v====v==+
539             #         |              |
540             #         |    server    |
541             #         |              |
542             #      +----#-----+--+
543             #      optional#     |
544             #      #########     |
545             #      #             |
546      +==+=v===v=+==+            |
547      |  | agent |  |            |
548      |  +-------+  |            |
549      |   PRINTER   <---------+
550      |             | Print Job Delivery Channel
551      |             |
552      +=============+
```
553   **Figure 3-3 - Configuration 3 - client-server-printer - client monitors printer agent**
554   **and server**
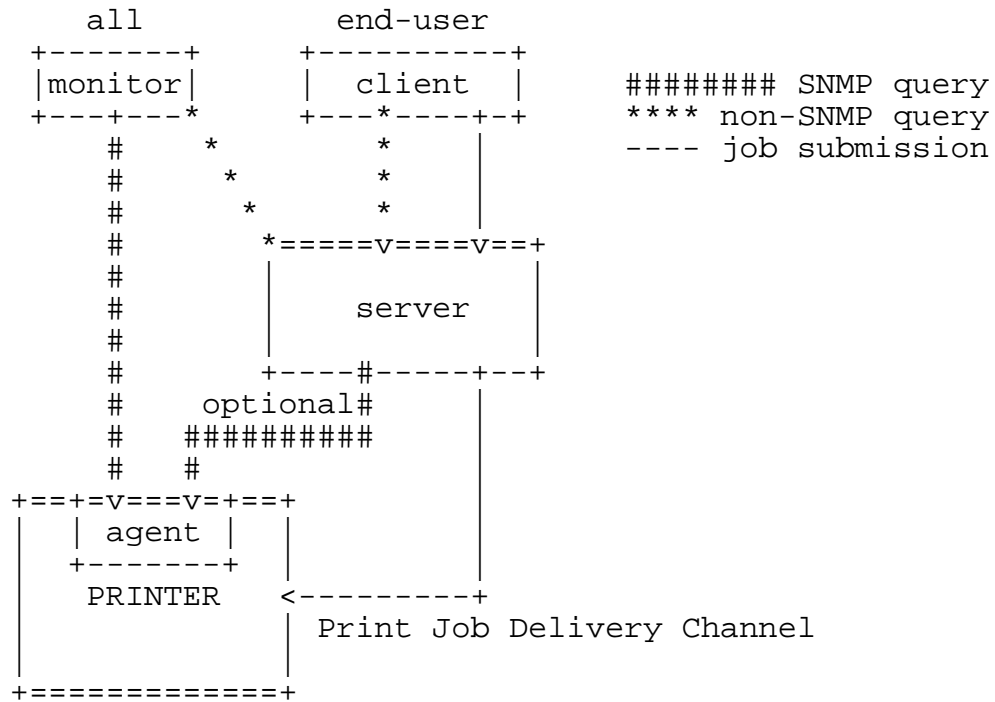
555   The Job Monitoring MIB is designed to support the following relationships (not shown in
556   Figure 3-3):
557       1.   Multiple **clients** MAY submit jobs to a **server**.
558       2.   Multiple **clients** MAY monitor a **server**.
559       3.   Multiple **monitors** MAY monitor a **server**.
560       4.   A **client** MAY submit jobs to multiple **servers**.
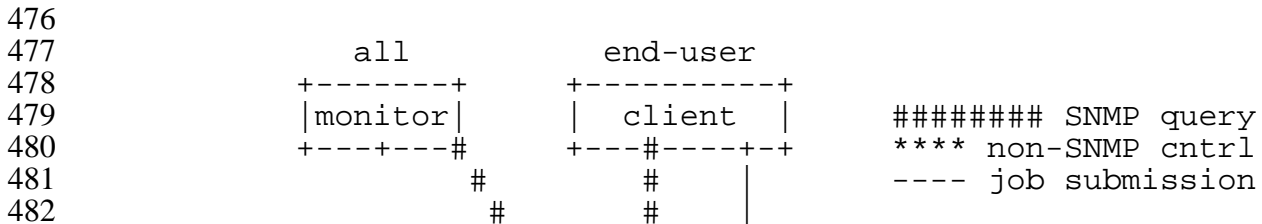561       5.   A **monitor** MAY monitor multiple **servers**.
562       6.   Multiple **servers** MAY submit jobs to a **printer**.
563       7.   Multiple **servers** MAY control a **printer**.

564    ## 4. Conformance Considerations

565    In order to achieve interoperability between job monitoring applications and job
566    monitoring agents, this specification includes the conformance requirements for both
567    monitoring applications and agents.

568    ### 4.1  Conformance Terminology

569    This specification uses the verbs: "SHALL", "SHOULD", "MAY", and "NEED NOT" to
570    specify conformance requirements according to RFC 2119 as follows:

571    - "SHALL":  indicates an action that the subject of the sentence must implement in
572      order to claim conformance to this specification

573    - "MAY":  indicates an action that the subject of the sentence does not have to
574      implement in order to claim conformance to this specification, in other words that
575      action is an implementation option

576    - "NEED NOT":  indicates an action that the subject of the sentence does not have to
577      implement in order to claim conformance to this specification.  The verb "NEED
578      NOT" is used instead of "may not", since "may not" sounds like a prohibition.

579    - "SHOULD":  indicates an action that is recommended for the subject of the
580      sentence to implement, but is not required, in order to claim conformance to this
581      specification.

582    ### 4.2  Agent Conformance Requirements

583    A conforming agent:

584    1.  SHALL implement *all* MANDATORY groups and attributes in this specification.

585    2.  SHALL implement *each* CONDITIONALLY MANDATORY attribute, if the server
586      or device that the agent is instrumenting has the feature represented by the
587      CONDITIONALLY MANDATORY attribute..

588    2.  NEED NOT implement any OPTIONAL attributes, whether the agent is able to obtain
589      the information from the server or device.

590    3.  NEED NOT implement both forms of an time attribute if it implements an time
591      attribute that permits a choice of Integer and Octets forms, though implementing both
592      forms may help management applications by giving them a choice of representations,
593      since the representation are equivalent.and is recommended *not* to provide both forms
594      for a particular time attribute.  See page 56.

595      NOTE - This MIB, like the Printer MIB, is written following the subset of SMIv2 that
596      can be supported by SMIv1 and SNMPv1 implementations.

597    **4.2.1  MIB II System Group objects**

598    The Job Monitoring MIB agent SHALL implement all objects in the system group of
599    MIB-II (RFC 1213)[5], whether the Printer MIB[1] is implemented or not.


600    **4.2.2  MIB II Interface Group objects**

601    The Job Monitoring MIB agent SHALL implement all objects in the Interfaces Group of
602    MIB-II (RFC 1213)[5], whether the Printer MIB[1] is implemented or not.


603    **4.2.3  Printer MIB objects**

604    If the agent is instrumenting a device that is a printer, the agent SHALL implement all of
605    the mandatory objects in the Printer MIB[1] and all the objects in other MIBs that
606    conformance to the Printer MIB requires, such as the Host Resources MIB (RFC
607    1514)[6].  If the agent is instrumenting a server that controls one or more networked
608    printers, the agent NEED NOT implement the Printer MIB and NEED NOT implement
609    the Host Resources MIB.


610    **4.3  Job Monitoring Application Conformance Requirements**

611    A conforming job monitoring application:

612    1.  SHALL accept all objects in all MANDATORY groups and all MANDATORY and
613        CONDITIONALLY MANDATORY attributes that are required to be implemented by
614        an agent according to Section 4.2 and SHALL either present them to the user or
615        ignore them.

616    2.  SHALL accept *all* OPTIONAL attributes, including enum and bit values specified in
617        this specificationstandard and additional ones that may be registered with IANA and
618        SHALL either present them to the user or ignore them.  In particular, a conforming
619        job monitoring application SHALL not malfunction when receiving any standard or
620        registered enum or bit values.  See Section 7 entitled "IANA Considerations" on page
621        21.

622    3.  SHALL accept either form of time attribute, if it supports a time attribute, since agents
623        are free to implement either time form.  See page 56.


624    **5.  Job Identification**

625    There are a number of attributes that permit a user, operator or system administrator to
626    identify jobs of interest, such as **jobOwner**, **jobName**, etc.  In addition, there is a Job
627    Submission ID object that allows a monitoring application to quickly locate and identify a
628    particular job of interest that was submitted from a particular client by the user invoking

629  the monitoring application.  The Job Monitoring MIB needs to provide for identification
630  of the job at both sides of the job submission process.  The primary identification point is
631  the client side.  The Job Submission ID allows the monitoring application to identify the
632  job of interest from all the jobs currently "known" by the server or device.  The Job
633  Submission ID can be assigned by either the client's local system or a downstream server
634  or device.  The point of assignment will be determined by the job submission protocol in
635  use.

636  The server/device-side identifier, called the **jmJobIndex** object, will be assigned by the
637  server or device that accepts the jobs from submitting clients.  The MIB agent SHALL use
638  the job identifier assigned by the server or device to the job as the value of the
639  **jmJobIndex** object that defines the table rows (there are multiple tables) that contain the
640  information relating to the job.  This object allows the interested party to obtain all objects
641  desired that relate to this job.  The MIB provides a mapping table that maps each Job
642  Submission ID to the corresponding **jmJobIndex** value, so that an application can
643  determine the correct value for the jmJobIndex value for the job of interest in a single Get
644  operation.  See the **jmJobIDGroup** on page 77.

645  The **jobName** attribute provides a name that the user supplies a<u>s</u><s>n</s> a job attribute with the
646  job.  The **jobName** attribute is not necessarily unique, even for one user, let alone across
647  users.


# 6. Internationalization Considerations

649  There are a number of objects in this MIB that are represented as coded character sets.
650  The data type for such objects is **OCTET STRING**.  Such objects could be in different
651  coded character sets and could be localized in the language and country, i.e., could be
652  localized.  However, for the Job Monitoring MIB, most of the objects are supplied as job
653  attributes by the client that submits the job to the server or device and so are represented
654  in the coded character set specified by that client.  Therefore, the agent is *not* able to
655  provide for different representations depending on the locale of the server, device, or user
656  of the job monitoring application.  The only exception is job submission protocols that
657  pass job or document attributes as OBJECT IDENTIFIERS or enums.  For those job and
658  document attributes, the agent SHALL represent the corresponding objects in the Job
659  Monitoring MIB as coded character sets in the current (default) locale of the server or
660  printer as established by the system administrator or the implementation.

661  For simplicity, this specification assumes that the clients, job monitoring applications,
662  servers, and devices are all running in the same locale.  However, this specification allows
663  them to run in any locale, including locales that use two-octet coded character sets, such
664  as ISO 10646 (Unicode).  Job monitors applications are expected to understand the coded
665  character set of the client (and job), server, or device.  No special means is provided for
666  the monitor to discover the coded character set used by jobs or by the server or device.

667 This specification does *not* contain an object that indicates what locale the server or device
668 is running in, let alone contain an object to control what locale the agent is to use to
669 represent coded character set objects.

670 This MIB also contains objects that are represented using the **DateAndTime** textual
671 convention from SNMPv2-TC (RFC 1903).  The job management application SHALL
672 display such objects in the locale of the user running the monitoring application.

## 673 7.  IANA Considerations

674 During the development of this standard, the Printer Working Group (PWG) working with
675 IANA will register additional enums while the standard is in the proposed and draft states
676 according to the procedures described in this section.  IANA will handle registration of
677 additional enums after this standard is approved in cooperation with an IANA-appointed
678 registration editor from the PWG according to the procedures described in this section:

### 679 7.1  IANA Registration of enums

680 This specification uses textual conventions to define enumerated values (enums) and bit
681 values.  Enumerations (enums) and bit values are sets of symbolic values defined for use
682 with one or more objects or attributes.  All enumeration sets and bit value sets are
683 assigned a symbolic data type name (textual convention).  As a convention the symbolic
684 name ends in "**TC**" for textual convention.  These enumerations are defined at the
685 beginning of the MIB module specification.

686 This working group has defined several type of enumerations for use in the Job
687 Monitoring MIB and the Printer MIB[1].  These types differ in the method employed to
688 control the addition of new enumerations.  Throughout this document, references to "type
689 n enum", where n can be 1, 2 or 3 can be found in the various tables.  The definitions of
690 these types of enumerations are:

#### 691 7.1.1  Type 1 enumerations

692 Type 1 enumeration:  All the values are defined in the Job Monitoring MIB specification
693 (RFC for the Job Monitoring MIB).  Additional enumerated values require a new RFC.

694 NOTE - There are no type 1 enums in the current draft.

#### 695 7.1.2  Type 2 enumerations

696 Type 2 enumeration:  An initial set of values are defined in the Job Monitoring MIB
697 specification.  Additional enumerated values are registered after review by this working
698 group. The initial versions of the MIB will contain the values registered so far. After the

699  MIB is approved, additional values will be registered through IANA after approval by this
700  working group.

701  The following type 2 enums are contained in the current draft :
702      **1.  JmTimeStampTC**
703      **2.  JmFinishingTC**
704      **3.  JmPrintQualityTC**
705      **4.  JmTonerEconomyTC**
706      **5.  JmPrinterResolutionTC**
707      **6.  JmTonerDensityTC**
708      **6.  JmMediumTypeTC**
709      **7.  JmJobStateTC**
710      **8.  JmAttributeTypeTC**

711  **7.1.3  Type 3 enumeration**

712  Type 3 enumeration:  An initial set of values are defined in the Job Monitoring MIB
713  specification.  Additional enumerated values are registered without working group review.
714  The initial versions of the MIB will contain the values registered so far.  After the MIB is
715  approved, additional values will be registered through IANA without approval by this
716  working group.

717  NOTE - There are no type 3 enums in the current draft.

718  **7.2  IANA Registration of type 2 bit values**

719  This draft contains the following type 2 bit value textual-conventions:
720      **1.  JmJobServiceTypesTC**
721      2.  **JmJobStateReasons1TC**
722      3.  **JmJobStateReasons2TC**
723      4.  **JmJobStateReasons3TC**
724      5.  **JmJobStateReasons4TC**
725  These textual-conventions are defined as bits in an Integer so that they can~~may~~ be used
726  with SNMPv1 SMI.  The **jobStateReasons***n* (*n*=1..4) attributes are defined as bit values
727  using the corresponding **JmJobStateReasons***n***TC** textual-conventions.

728  The registration of **JmJobServiceTypesTC** and **JmJobStateReasons***n***TC** bit values
729  SHALL follow the procedures for a type 2 enum as specified in Section 7.1.2.

730  **7.3  IANA Registration of Job Submission Id Formats**

731  In addition to enums and bit values, this specification assigns numbers to various job
732  submission ID formats.  See **jmJobSubmissionID** on page 78.  The registration of

733  **jmJobSubmissionID** format numbers SHALL follow the procedures for a type 2 enum as
734  specified in Section 7.1.2.


## 8.  Security Considerations

### 8.1  Read-Write objects

737  All objects are read-only greatly simplifying the security considerations.  If another MIB
738  augments this MIB, that MIB might allow objects in this MIB to be modified.  However,
739  that MIB SHALL have to support the required access control in order to achieve security,
740  not this MIB.


### 8.2  Read-Only Objects In Other User's Jobs

742  The security policy of some sites may be that unprivileged users can only get the objects
743  from jobs that they submitted, plus a few minimal objects from other jobs, such as the
744  **jmJobKOctetsRequested** and **jmJobKOctetsCompleted** objects**,** so that a user can tell
745  how busy a printer is.  Other sites might allow all unprivileged users to see all objects of
746  all jobs.  It is up to the agent to implement any such restrictions based on the identification
747  of the user making the SNMP request.  This MIB does not require, nor does it specify
748  how, such restrictions would be implemented.  A monitoring application SHOULD
749  enforce the site security policy with respect to returning information to an unprivileged
750  end user that is using the monitoring application to monitor jobs that do not belong to that
751  user, i.e., the **jobOwner** attribute in the **jmAttributeTable** does not match the user's user
752  name.  See the **JmAttributeTypeTC** textual convention on page 53 and the
753  **jmAttributeTable** on page 86.

754  An operator is a privileged user that would be able to see all objects of all jobs,
755  independent of the policy for unprivileged users.


## 9.  Returning Objects With No Value In Mandatory Groups

757  If an object in a mandatory group does not have an instrumented value for a particular job
758  submission protocol or the job submitting client did not supply a value (and the accepting
759  server or device does not supply a default), this MIB requires that the agent SHALL
760  follow the normal SNMP practice of returning a distinguished value, such as a zero-length
761  string, an **unknown(2)** value for an enum, or a (**-2**) for an integer value.


## 10.  Notification and Traps

763  This MIB does not specify any traps.  For simplicity, management applications are
764  expected to poll for status.  The resulting network traffic is not expected to be significant.

765    **11. MIB specification**

766    The following pages constitute the actual Job Monitoring MIB.

```
767   Job-Monitoring-MIB DEFINITIONS ::= BEGIN
768
769   IMPORTS
            MODULE-IDENTITY, OBJECT-TYPE, experimental,
            Integer32                                   FROM SNMPv2-SMI
            TEXTUAL-CONVENTION                          FROM SNMPv2-TC
            MODULE-COMPLIANCE, OBJECT-GROUP             FROM SNMPv2-CONF;
            -- The following textual-conventions are needed
            -- to implement certain attributes, but are not
            -- needed to compile this MIB.  They are
            -- provided here for convenience:
            -- DateAndTime                              FROM SNMPv2-TC
            -- PrtAlertCodeTC, PrtInterpreterLangFamilyTC   FROM Printer-MIB
770
771   -- Use the experimental (54) OID assigned to the Printer MIB[1] before
772   -- it was published as RFC 1759.
773   -- Upon publication of the Job Monitoring MIB as an RFC, delete this
774   -- comment and the line following this comment and change the
775   -- reference of { temp 104 } (below) to { mib-2 X }.
776   -- This will result in changing:
777   -- 1 3 6 1 3 54 jobmonMIB(105)     to:
778   -- 1 3 6 1 2  1 jobmonMIB(X)
779   -- This will make it easier to translate prototypes to
780   -- the standard namespace because the lengths of the OIDs won't
781   -- change.
782   temp OBJECT IDENTIFIER ::= { experimental 54 }
783
784   jobmonMIB MODULE-IDENTITY
785       LAST-UPDATED "9705420240000Z"
786       ORGANIZATION "IETF Printer MIB Working Group"
787       CONTACT-INFO
788           "Tom Hastings
789           Postal:  Xerox Corp.
790                    Mail stop ESAE-231
791                    701 S. Aviation Blvd.
792                    El Segundo, CA 90245
793
794           Tel:     (301)333-6413
795           Fax:     (301)333-5514
796           E-mail:  hastings@cp10.es.xerox.com"
797       DESCRIPTION
798           "The MIB module for monitoring job in servers, printers, and
799           other devices.
800
801           File: jmp-mib.doc,  .pdf,  .txt,  .mib
802           Version: 0.821"
803       ::= { temp 105 }
804
805
```

```
806
807    -- Textual conventions for this MIB module
808
809
810    JmTimeStampTC ::= TEXTUAL-CONVENTION
811        STATUS      current
812        DESCRIPTION
813            "The simple time at which an event took place.  The units SHALL
814            be in seconds since the system was booted.
815
816            NOTE - JmTimeStampTC is defined in units of seconds, rather than
817            100ths of seconds, so as to be simpler for agents to implement
818            (even if they have to implement the 100ths of a second to comply
819            with implementing sysUpTime in MIB-II[5].)
820
821            NOTE - JmTimeStampTC is defined as an Integer32 so that it can
822            be used as a value of an attribute, i.e., as a value of the
823            jmAttributeValueAsInteger object (see page 88).  The TimeStamp
824            textual-convention defined in SMNPv2-TC is defined as an
825            APPLICATION 3 IMPLICIT INTEGER tag, not an Integer32, so cannot
826            be used in this MIB as one of the values of
827            jmAttributeValueAsInteger."
828        SYNTAX      INTEGER(0..2147483647)
829
830
831
832
833    JmJobSourcePlatformTypeTC ::= TEXTUAL-CONVENTION
834        STATUS      current
835        DESCRIPTION
836            "The source platform type that can submit jobs to servers or
837            devices in any of the 3 configurations."
838
839        -- This is a type 2 enumeration.  See Section 7.1 on page 21.
840        SYNTAX      INTEGER {
               other(1),
               unknown(2),
               sptUNIX(3),            -- UNIX(tm)
               sptOS2(4),             -- OS/2
               sptPCDOS(5),           -- DOS
               sptNT(6),              -- NT
               sptMVS(7),             -- MVS
               sptVM(8),              -- VM
               sptOS400(9),           -- OS/400
               sptVMS(10),            -- VMS
               sptWindows95(11),      -- Windows95
               sptNetWare(33)         -- NetWare
```

```
841        }
842
843
844
845
846
847    JmFinishingTC ::= TEXTUAL-CONVENTION
848        STATUS       current
849        DESCRIPTION
850            "The type of finishing."
851
852        -- This is a type 2 enumeration.  See Section 7.1 on page 21.
853        SYNTAX       INTEGER {
               other(1),
                   -- Some other finishing besides one of the specified or
                   -- registered values.

               unknown(2),
                   -- The finishing is unknown.

               none(3),
                   -- Perform no finishing.

               staple(4),
                   -- Bind the document(s) with one or more staples. The
                   -- exact number and placement of the staples is site-
                   -- defined.

               stapleTopLeft(5),
                   -- Place one or more staples on the top left corner of
                   -- the document(s).

               stapleBottomLeft(6),
                   -- Place one or more staples on the bottom left corner
                   -- of the document(s).

               stapleTopRight(7),
                   -- Place one or more staples on the top right corner of
                   -- the document(s).

               stapleBottomRight(8),
                   -- Place one or more staples on the bottom right corner
                   -- of the document(s).

               saddleStitch(9),
                   -- Bind the document(s) with one or more staples (wire
                   -- stitches) along the middle fold.  The exact number
                   -- and placement of the stitches is site-defined.
```

```
        edgeStitch(10),
            -- Bind the document(s) with one or more staples (wire
            -- stitches) along one edge.  The exact number and
            -- placement of the staples is site-defined.

        punch(11),
            -- This value indicates that holes are required in the
            -- finished document. The exact number and placement of
            -- the holes is site-defined  The punch specification
            -- MAY be satisfied (in a site- and implementation-
            -- specific manner) either by drilling/punching, or by
            -- substituting pre-drilled media.

        cover(12),
            -- This value is specified when it is desired to select
            -- a non-printed (or pre-printed) cover for the
            -- document. This does not supplant the specification of
            -- a printed cover (on cover stock medium) by the
            -- document itself.

        bind(13)
            -- This value indicates that a binding is to be applied
            -- to the document; the type and placement of the
            -- binding is site-defined.
854     }
855
856
857
858
859
860 JmPrintQualityTC ::= TEXTUAL-CONVENTION
861     STATUS      current
862     DESCRIPTION
863         "Print quality settings."
864
865     -- This is a type 2 enumeration.  See Section 7.1 on page 21.
866     SYNTAX      INTEGER {
            other(1),    --  Not one of the specified or registered
                         --  values.
            unknown(2),  --  The actual value is unknown.
            draft(3),    --  Lowest quality available on the printer.
            normal(4),   --  Normal or intermediate quality on the
                         --  printer.
            high(5)      --  Highest quality available on the printer.
867     }
868
```

```
869
870   JmPrinterResolutionTC ::= TEXTUAL-CONVENTION
871       STATUS        current
872       DESCRIPTION
873           "Printer resolutions.
874
875           The values are type2 enums that represent single integers or
876           pairs of integers.  The latter are to specify the resolution
877           when the x and y dimensions differ.  When two integers are
878           specified, the first is in the x direction, i.e., in the
879           direction of the shortest dimension of the medium, so that the
880           value is independent of whether the printer feeds long edge or
881           short edge first."
882
883       -- This is a type 2 enumeration.  See Section 7.1 on page 21.
884       SYNTAX        INTEGER {
              other(1),              -- Not one of the specified or registered
                                     -- values.
              unknown(2),           -- The actual value is unknown.
              normal(3),            -- Normal resolution.
              res100(4),            -- 100 x 100 dpi
              res200(5),            -- 200 x 200 dpi
              res240(6),            -- 240 x 240 dpi
              res300(7),            -- 300 x 300 dpi
              res360(8),            -- 360 x 360 dpi
              res600(9),            -- 600 x 600 dpi
              res720(10),           -- 720 x 720 dpi
              res800(11),           -- 800 x 800 dpi
              res1200(12),          -- 1200 x 1200 dpi
              res1440(13),          -- 1440 x 1440 dpi
              res1800(14),          -- 1800 x 1800 dpi
              res100x200(15),       -- 100 x 200 dpi
              res300x600(16),       -- 300 x 600 dpi
              res600x300(17),       -- 600 x 300 dpi
              res360x720(18),       -- 360 x 720 dpi
              res720x360(19),       -- 720 x 360 dpi
              res400x800(20),       -- 400 x 800 dpi
              res800x400(21),       -- 800 x 400 dpi
              res600x1200(22),      -- 600 x 1200 dpi
              res1200x600(23),      -- 1200 x 600 dpi
              res720x1440(24),      -- 720 x 1440 dpi
              res1440x720(25),      -- 1440 x 720 dpi
              res1800x600(26)       -- 1800 x 600 dpi
885       }
886
887
888
889
```

```
890
891    JmTonerEconomyTC ::= TEXTUAL-CONVENTION
892        STATUS      current
893        DESCRIPTION
894            "Toner economy settings."
895
896        -- This is a type 2 enumeration.  See Section 7.1 on page 21.
897        SYNTAX      INTEGER {
               off(0),        -- Off.  Normal.  Use full toner.
               on(1)          -- On.  Use less toner than normal.
898        }
899
900
901
902
903
904    JmMediumTypeTC ::= TEXTUAL-CONVENTION
905        STATUS      current
906        DESCRIPTION
907            "Identifies the type of medium."
908
909        -- This is a type 2 enumeration.  See Section 7.1 on page 21.
910        SYNTAX      INTEGER {
         other(1),
               -- The type is neither one of the values listed in this
               -- specification nor a registered value.

         unknown(2),
               -- The type is not known.

         stationery(3),
               -- Separately cut sheets of an opaque material.

         transparency(4),
               -- Separately cut sheets of a transparent material.

         envelope(5),
               -- Envelopes that can be used for conventional mailing
               -- purposes.

         envelopePlain(6),
               -- Envelopes that are not preprinted and have no windows.

         envelopeWindow(7),
               -- Envelopes that have windows for addressing purposes.

         continuousLong(8),
```

```
                 -- Continuously connected sheets of an opaque material
                 -- connected along the long edge.

           continuousShort(9),
                 -- Continuously connected sheets of an opaque material
                 -- connected along the short edge.

           tabStock(10),
                 -- Media with tabs.

           multiPartForm(11),
                 -- Form medium composed of multiple layers not pre-attached
                 -- to one another;  each sheet MAY be drawn separately from
                 -- an input source.

           labels(12),
                 -- Label-stock.

           multiLayer(13)
                 -- Form medium composed of multiple layers which are pre-
                 -- attached to one another, e.g. for use with impact
                 -- printers.
```
911       `}`
912
913
914
915
916
917   **JmJobStateTC** ::= TEXTUAL-CONVENTION
918       STATUS        current
919       DESCRIPTION
920           "The current state of the job (**pending**, **processing**,
921           ~~**completed**~~**held**, etc.)~~.~~
922
923           ~~Management applications shall be prepared to receive all the~~
924           ~~standard job states.  Agents instrumenting servers and devices~~
925           ~~are not required to generate all job states, only those that are~~
926           ~~indicated as 'mandatory' in the enum definitions below.  The~~
927           ~~remaining job states are 'conditionally mandatory', i.e., an~~
928           ~~agent for a server or device shall implement each of the~~
929           ~~remaining states if server or device jobs have states with the~~
930           ~~same semantics.  See Section  entitled '' on page for additional~~
931           ~~job state semantics, legal job state transitions, and~~
932           ~~implementation considerations.~~
933
934           ~~Companion textual conventions (**JmJobStateReasonsnTC, n=1..4**) and~~
935           ~~corresponding attributes (**jobStateReasonsn**) provide additional~~

936  ~~information about job states.  While the job states cannot be~~
937  ~~added to without impacting deployed clients that take actions~~
938  ~~upon receiving job state values, it is the intent that~~
939  ~~additional **JmJobStateReasonsnTC** enums can be defined without~~
940  ~~impacting deployed clients.  In other words, the~~
941  ~~**JmJobStateReasonsnTC** TCs are intended to be extensible.  See~~
942  ~~page .~~
943
944  ~~The following job state standard values are defined:~~The
945  following figure shows the normal job state transitions:
946
947                                              +----> canceled(7)
948                                             /
949      +----> pending(3) ---------> processing(5) -------+------> completed(9)
950      |          ^                      ^               \
951  --->+          |                      |                +----> aborted(8)
952      |          v                      v               /
953      +----> pendingHeld(4)   processingStopped(6) ----+
954
955  ~~<---------------- active ----------------->   <----- in-active --->~~
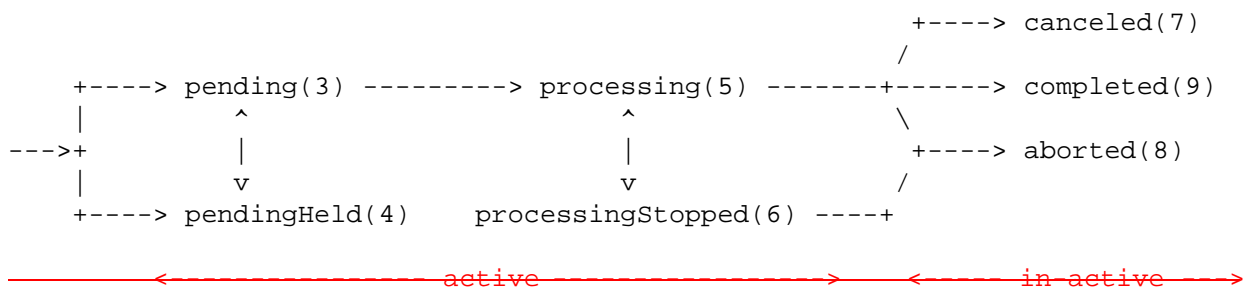
**Figure 4 - Normal Job State Transitions**

957
958      Normally a job progresses from left to right.  Other state
959      transitions are unlikely, but are not forbidden.  Not shown are
960      the transitions to ~~T~~the **canceled** state ~~can be entered~~ from the
961      **pending**, **pendingHeld**, **processing**, and~~or~~ **processingStopped**
962      states.
963
964      Jobs in the **pending**, **processing**, and **processingStopped** states
965      are called 'active', while jobs in the **pendingHeld**, **canceled**,
966      **aborted**, and **completed** are called 'in-active'."
967
968      -- This is a type 2 enumeration.  See Section 7.1 on page 21.
969      SYNTAX      INTEGER {
         **other(1),**
             -- The job state is *not* one of the defined states.

         **unknown(2),**
             -- The job state is *not* known, or its state is
             -- indeterminate.

         **pending(3~~4~~),**
             -- The job is a candidate to start~~a candidate for~~
             -- processing, but is not yet processing.

         **pending~~H~~held(4~~3~~),**
             -- The job is not ~~yet~~ a candidate for processing for
             -- any number of reasons but will return to the

```
        -- pending state as soon as the reasons are no longer
        -- present.  The job's jmJobStateReasons1 object
        -- and/or jobStateReasonsn (n=2..4) attributes SHALL
        -- indicate why the job is no longer a candidate for
        -- processing.  The reasons are represented as bits
        -- in the jmJobStateReasons1 object and/or
        -- jobStateReasonsn (n=2..4) attributes. Some
        -- reasons are used in other states to give added
        -- information about the job state.  See the
           JmJobStateReasonsn1TC (n=1..4) textual convention
           on page (60) for the specification of each reason
           and in which states the reasons are intended to be
           used.

    processing(55),                          MANDATORY
        -- Either:
        --
        -- 1.  The job is using, or is attempting to use, one
        -- or more document transforms which include (1)
        -- purely software processes that are, such as
        -- interpreting a PDL, and (2) hardware devices that
        -- are interpreting a PDL, , but is not yet making
        -- marks on a medium, and/or performing finishing,
        -- such as stapling, etc.
        --
        -- OR
        --
        -- 2. (configuration 2) the server has made the job
        -- ready for printing, but the output device is not
        -- yet printing it, either because the job hasn't
        -- reached the output device or because the job is
        -- queued in the output device or some other spooler,
        -- awaiting the output device to print it.
        --
        --
        -- If an implementation does not distinguish between
        -- processing and printing, then the processing state
        -- shall be implemented.
        -- When the job is in the processing state, the
        -- entire job state includes the detailed status
        -- represented in the device MIB indicated by the
        -- hrDeviceIndex value of the job's physicalDevice
        -- attribute, if the agent implements such a device
        -- MIB.
        --
        -- Implementations MAY, though they NEED NOT, include
           additional values in the job's jmJobStateReasons1
           object to indicate the progress of the job, such
           as adding the jobPrinting value to indicate when
```

the device is actually making marks on a medium.

~~printing(6),~~
    ~~-- The job is printing, i.e., making marks on a~~
    ~~-- medium.~~
    ~~--~~
    ~~-- If an implementation does *not* distinguish between~~
    ~~**processing** and **printing**, then the **processing** state~~
    ~~shall be implemented.~~

**processingStopped~~needsAttention~~(6~~7~~                ~~MANDATORY~~
),**
    -- The job has stopped while processing for any
    -- number of reasons and will return to the
    -- **processing** state as soon as the reasons are no
    -- longer present.
    --
    -- The job's **jmJobStateReasons1** object and/or the
    -- job's **jobStateReasons*n*** (***n*=2..4**) attributes MAY
    -- indicate why the job has stopped processing.  ~~is~~
    ~~-- using one or more devices, but has encountered a~~
    ~~-- problem with at least one device that requires~~
    ~~-- human intervention before the job can continue~~
    ~~-- using that device.  Examples include running out~~
    ~~-- of paper or a paper jam.~~
    --
    -- For example, if the output device is stopped, the
    -- **device~~Stopped~~** value MAY be included in the job's
    -- **jmJobStateReasons1** object.
    --
    -- NOTE - When an output device is stopped, the
    -- device~~s~~ usually indicate its~~their~~ condition in
    human readable form locally at the device.  The
    management application can obtain more complete
    device status remotely by querying the appropriate
    device MIB using the job's **deviceIndex**
    attribute(s), if the agent implements such a
    device MIB~~.~~

**canceled(7~~8~~),**                        ~~MANDATORY~~
    -- A client has canceled the job and ~~T~~the job is
    -- either: (1) in the process of being terminated by
    -- the server or device or (2) has completed
    -- terminating~~the job, either because the client~~
    ~~-- canceled the job or because a serious problem was~~
    ~~-- encountered by a document transform while~~
    ~~processing the job~~.  The job's **jmJobStateReasons1**
    object~~attribute~~ SHOULD~~shall~~ contain either the
    **canceledByUser** or **canceledByOperator** value~~the~~

        reasons that the job was canceled.  The job shall
        remain in the **canceled** state for the same period
        of time as if the job had completed, before
        transiting to the **unknown** state.  See the
        **completed** state description.

    **aborted(8),**
        -- The job has been aborted by the system, usually
        -- while the job was in the **processing** or
        -- **processingStopped** state.

    **completed(9)**                          MANDATORY
        -- The job has (1) completed successfully or with
        -- warnings or errors after processing/printing and
        -- all of the media have been successfully stacked in
        -- the appropriate output bin(s).
        --
        -- The job has completed successfully or with
        -- warnings or errors.  The job's **jmJobStateReasons1**
        -- objectattribute SHOULDshall contain one of:
        -- **completedSuccessfully**, **completedWithWarnings**, or
        -- **completedWithErrors** valuesthe reasons that the job
        -- has entered the **completed** state.
        --
            The length of time that a job may be in the
            **completed** state, before transitioning to **unknown**,
            is specified by the value of the
            **jmGeneralJobPersistence** object.  In addition, the
            agent shall maintain all of the attributes in the
            **jmAttributeTable** for at least the time specified
            in the **jmGeneralAttributePersistence** object, so
            that a management application accounting program
            can copy all the attributes to an accounting log.

970     }
971
972
973   **JmAttributeTypeTC** ::= TEXTUAL-CONVENTION
974       STATUS        current
975       DESCRIPTION
976           "The type of the attribute which identifies the attribute.
977
978           Some attributes represent information about a job, such as a
979           file-name, or a document-name, or submission-time or completion
980           time.  Other attributes represent resources required, e.g., a
981           medium or a colorant, etc. to process the job before the job
982           start processing OR to indicate the amount of the resource that
983           is being consumed while the job is processing, e.g., pages

984         completed or impressions completed.  If both a required and a
985         consumed value of a resource is needed, this specification
986         assigns two separate attribute enums in the textual convention.
987
988         Most attributes apply to all three configurations covered by
989         this MIB specification (see section 3 on page 14).  Those
990         attribute that apply to a particular configuration are indicated
991         as '**Configuration *n*:**'.
992
993         **Conformance of Attribute Implementation**
994
995         A very few Some attributes are MANDATORY for conformance, and
996         the rest are OPTIONALCONDITIONALLY MANDATORY.  An agent SHALL
997         instrument any MANDATORY attribute.  If the server or device
998         does not provide access to the information about the MANDATORY
999         attribute, the agent SHALL return the **'unknown'** value.  For
1000        attributes represented by a counting integer, the unknown value
1001        is (**-2**) and for attributes represented by an enum, the unknown
1002        value is (**2**), as in the Printer MIB[1].  For attributes
1003        represented by an OCTET STRING, the unknown value is a zero-
1004        length string, unless specified otherwise.
1005
1006        An agent shall instrument any CONDITIONALLY MANDATORY attribute
1007        if the server or device provides access to the information about
1008        the attribute to the agent.  If the server or device does *not*
1009        provide access to the information about the CONDITIONALLY
1010        MANDATORY attribute, the agent need *not* create the row in the
1011        **jmAttributeTable.**
1012
1013        The mandatory attributes are the ones required to have copies in
1014        the **jmJobStateTable** and to remain in the **jmAttributeTable**
1015        longer.  The MANDATORY attributes are:
1016
1017             **jobOwner(2015)**
1018
1019        The attributes not labeled as MANDATORY are OPTIONAL.  An agent
1020        MAY, but NEED NOT, implement any OPTIONAL attributes.
1021
1022        NOTE – The table of contents lists all the attributes in order
1023        to help see the order of enum assignments which is the order
1024        that the GetNext operation can be used to get attributes.  The
1025        table of contents also indicates the MANDATORY attributes as:
1026        (MANDATORY).
1027
1028        NOTE – The enum assignments are grouped logically with values
1029        assigned in groups of 20, so that additional values may be
1030        registered in the future and assigned a value that is part of
1031        their logical grouping.
1032

1033  ~~Attribute Creation~~

1034

1035  ~~An agent shall create a row in the **jmAttributeTable** for each~~
1036  ~~attribute that is (1) supplied with a job when the job is~~
1037  ~~accepted by a server or device or that (2) the server or device~~
1038  ~~supplies as a default either when the job is accepted or later~~
1039  ~~during processing.  The agent SHALL create the MANDATORY~~
1040  ~~attributes when the job is accepted.  The agent MAY create the~~
1041  ~~remaining attributes when the agent has the information.~~

1042

1043  **Datatypes and Attribute Naming Conventions**

1044

1045  The datatype of each attribute is indicated on the first line(s)
1046  of the description.  Some attributes have several different data
1047  type representations.  When the data types can be represented in
1048  a single row in the **jmAttributeTable**, the data type name is not
1049  included as the last part of the name of the attribute.  When
1050  the data types cannot be represented by a single row in the
1051  **jmAttributeTable**, ~~E~~each such representation is considered a
1052  separate attribute and is assigned a separate name and enum
1053  value.  For these attributes, the name of the datatype is the
1054  last part of the name of the attribute: **Name, Index,**
1055  **DateAndTime, TimeStamp,** etc.

1056

1057  NOTE: No attribute name exceeds 31 characters.

1058

1059  **Single-Value (Row) Versus Multi-Value (MULTI-ROW) Attributes**

1060

1061  Most attributes SHALL have only one row per job.  However, a few
1062  attributes can have multiple values per job or even per
1063  document, where each value is a separate row in the
1064  **jmAttributeTable**.  Unless indicated with '**MULTI-ROW:**' in
1065  **JmAttributeTypeTC**, an agent SHALL ensure that each attribute
1066  item occurs only once in the **jmAttributeTable** for a job.
1067  Attributes that are permitted to appear multiple times in the
1068  **jmAttributeTable** for a job are indicated with '**MULTI-ROW:**' in
1069  their specification in the **JmAttributeTypeTC**.  However, such
1070  '**MULTI-ROW**' attribute items SHALL *not* contain duplicates for
1071  'intensive' (as opposed to 'extensive') attributes.

1072

1073          For example, a job or document(s) may use multiple PDLs.
1074          However, each distinct **documentFormat**~~Type~~ attribute
1075          value ~~entry~~ SHALL appear in the **jmAttributeTable** only
1076          once for a job since the interpreter language is an
1077          intensive attribute item, even though the job has a
1078          number of documents that all use the same PDL.

1079

1080          As another example of an intensive attribute that can
1081          have multiple entries, if a document or job uses

1082          multiple types of media, there SHALL be only one row in
1083          the **jmAttributeTable** for each media type, not one row
1084          for each document that uses that medium type.
1085
1086          On the other hand, if a job contains two documents of
1087          the same name, there can be separate rows for the
1088          **documentName** attribute item with the same name, since a
1089          document name is an extensive attribute item.  The
1090          specification indicates that the values NEED NOT be
1091          unique for such '**MULTI-ROW:** attributes'

1093     **Value Represented As Integer Or Octets**

1095     In the following definitions of the enums, each description
1096     indicates whether the value of the attribute SHALL be
1097     represented using the **jmAttributeValueAsInteger** or the
1098     **jmAttributeValueAsOctets** objects by the initial tag: '**INTEGER:**'
1099     or '**OCTETS:**', respectively.  Some attributes allow the agent a
1100     choice of either an integer and/or an octets representation,
1101     depending on implementation.  These attributes are indicated
1102     with '**INTEGER:**' and/or '**OCTETS:**' tags.  A very few attributes
1103     require~~use~~ both objects at the same time to represent a pair of
1104     values (see **mediumConsumed~~Name~~(17165)**)~~ and so have both tags~~.
1105     These attributes are indicated with '**INTEGER:**' and/or '**OCTETS:**'
1106     tags.  See the **jmAttributeGroup** starting on page 85 for the
1107     descriptions of these objects.

1109     **Consumption Attributes**

1111     A number of attributes record consumption.  Such attribute names
1112     end with the word **'Completed'** or **'Consumed'**.  If the job has not
1113     yet consumed what that resource is metering, the agent either:
1114     (1) SHALL return the value **0** or (2) SHALL *not* add this attribute
1115     to the **jmAttributeTable** until the consumption begins.  In the
1116     interests of brevity, the semantics for **0** is specified once here
1117     and is *not* repeated for each **xxxxYyyyCompleted** and
1118     **xxxxYyyyConsumed** attribute specification.

1120     **Index Value Attributes**

1122     A number of attributes are indexes in other tables.  Such
1123     attribute names end with the word **'Index'**.  If the agent does
1124     not (yet) know the index value for a particular index attribute
1125     for a job, the agent either: (1) SHALL return the value **0** or (2)
1126     SHALL *not* add this attribute to the **jmAttributeTable** until the
1127     index value is known.  In the interests of brevity, the
1128     semantics for **0** is specified once here and is *not* repeated for
1129     each index attribute specification.
1130

```
1131        Attribute Naming Conventions
1132
1133        Attribute names often end in the data type, especially when
1134        there are more than one data type for the same information.
1135        Thus the suffixes are used: Name, Index, DateAndTime, TimeStamp,
1136        etc.
1137
1138        NOTE: No attribute name exceeds 31 characters.
1139
1140        The standard attribute types defined so far are:"
1141
1142    -- This is a type 2 enumeration.  See Section 7.1 on page 21.
1143    SYNTAX      INTEGER {
      -- jmAttributeTypeIndex              Datatype
        -- Description - including 'OCTETS:' or 'INTEGER:' to
        -- specify whether the value SHALL be represented in the
        -- jmAttributeValueAsOctets or the jmAttributeValueAsInteger
        -- object, or both, respectively.

      other(1),                         -- Integer32(-2..2147483647)
                                        -- AND/OR
                                        -- OCTET STRING(SIZE(0..63))
        -- INTEGER: and/or  OCTETS:  An attribute that is not in
        -- the list and/or that has not been approved and registered
        -- with IANA.

      unknown(2),                       -- Integer32(-2..2147483647)
                                        -- OR
                                        -- OCTET STRING(SIZE(0..63))
        -- INTEGER:  or  OCTETS:  An attribute whose semantics are
        -- not known to the agent.


      -- +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      -- Job State attributes
      --
      -- The following attributes specify the state of a job.
      -- +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

      jobState(3),                      -- JmJobStateTC (pg )
        -- INTEGER:  The current state of the job (pending,
        -- processing, held, etc.).  The final value for this
        -- attribute shall be either completed or canceled, before
        -- the agent removes the job from the table.
        --
        -- Management applications shall be prepared to receive all
        -- the standard job states.  Servers and devices are not
        -- required to generate all job states, only those which are
        -- appropriate for the particular implementation.
```

```
   --
   -- NOTE   Companion textual conventions,
   -- JmJobStateReasonsnTC (n=1..4 - see page ) and
   -- corresponding attributes ( - see page ) provides
   -- additional information about job states.  While the job
   -- states cannot be added to without impacting deployed
   -- clients, it is the intent that additional
   -- JmJobStateReasonsnTC enums can be defined without
   -- impacting deployed clients.

jobStateAssociatedValue(4),        -- Integer32(-2..2147483647)
   -- INTEGER:  The value of the most relevant attribute
   -- associated with the job's current state.
   --
   -- Which attribute depends on the job's current state (as
   -- specified by the value of the jmJobState/jobState
   -- object/attribute) as follows:
   --
   --    jmJobState        Associated Attribute            Page
   --    /jobState
   --
   --
   --
   -- NOTE   The jobStateAssociatedValue attribute selects from
   -- amongst seven mandatory attributes that attribute that is
   -- most relevant to the job's current state.  the
   -- jobStateAssociatedValue attribute is provided as an
   -- efficiency improvement, so that an application can obtain
   -- the most relevant attribute for each job's current state
   -- (1) without first having to determine the job's state or
   -- (2) having to request all seven mandatory attributes in
   -- the same GetNext operation that obtains the next job in
   -- the next conceptual row in the jmAttributeTable.
   --
   --
   --
   --
   --
   --
jobStateReasons1(5),               -- JmJobStateReasons1TC (pg )
   -- OCTETS:  Additional information about the job's current
   -- state that augments the jmJobState/jobState
   -- object/attribute.  The jobStateReasons1 attribute
   -- identifies the reason or reasons that the job is in the
   -- held, pending, processing, needsAttention, canceled, or
   -- completed state.  The agent shall indicate the particular
   -- reason(s) by setting the value of the jobStateReasons1
   -- attribute.  When the job does not have any reasons for
   -- being in its current state, the agent shall set the value
```

```
         -- of the jobStateReasons1 attribute to 0.
         --
         -- While the job states cannot be added to without impacting
         -- deployed clients, it is the intent that additional
         -- JmJobStateReasons1TC bit values can be defined without
         -- impacting deployed clients.  In other words, the
         -- JmJobStateReasons1TC TC is intended to be extensible.
         --
         -- Companion job state reasons TCs: JmJobStateReasons2TC,
         -- JmJobStateReasons3TC, JmJobStateReasons4TC, are
         -- defined/reserved for additional 31*3 = 93 job state
         -- reasons for use with the corresponding attributes:
         -- jobStateReasons2, jobStateReasons3, and jobStateReasons4.
         -- This is a type 2 bit definition.  See section  on page .
         --
jobStateReasons2(36),           -- JmJobStateReasons2TC (pg 64)
         -- INTEGEROCTETS:  Additional information about the job's
         -- current state that augments the jmJobState/jobState
         -- object/attribute.  See the description under the
         -- jJmJobStateReasons1TC textual-conventionattribute on page
            60.

            This is a type 2 bit definition.  See section  on page .

jobStateReasons3(47),           -- JmJobStateReasons3TC (pg 70)
         -- INTEGEROCTETS:  Additional information about the job's
         -- current state that augments the jmJobState/jobState
         -- object/attribute.  See the description under
         -- jJmJobStateReasons1TC textual-conventionattribute  on
            page 60.

jobStateReasons4(58),           -- JmJobStateReasons4TC (pg 70)
         -- INTEGEROCTETS:  Additional information about the job's
         -- current state that augments the jmJobState/jobState
         -- object/attribute.  See the description under
         -- jJmJobStateReasons1TC textual-conventionattribute  on
            page 60.

numberOfInterveningJobs(9),     -- Integer32(-2..2147483647)
         -- INTEGER:  The number of jobs that are expected to be
         -- processed before this job is processed according to the
         -- implementation's queuing algorithm if no other jobs were
         -- to be submitted.  In other words, this value is the job's
         -- queue position.  The agent shall return a value of 0 for
         -- this attribute when this job starts processing (since
         -- there are no jobs in front of the job).

deviceAlertCode(610),           -- PrtAlertCodeTC (Printer-MIB)
         -- INTEGER:  The device alert code when the job is stopped
```

```
            -- because the device needs attention, i.e., needs human
            -- intervention.  When the device is a printer, this device
            -- alert code SHALL be the printer alert code defined by the
            -- Printer MIB[1] using the PrtAlertCodeTC textual
            -- convention or equivalent.

   processingMessage(711),            -- OCTET STRING(SIZE(0..63))
            -- OCTETS:  MULTI-ROW:  A coded character set message that
            -- is generated during the processing of the job as a simple
            -- form of processing log to show progress and any problems.
            --
            -- There is no restriction on the same message in multiple
            -- rows.



   -- ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   -- Job Identification attributes
   --
   -- The following attributes help an end user, a system
   -- operator, or an accounting program identify a job.
   -- ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

   jobOwner(2015),                    -- OCTET STRING(SIZE(0..63))
                                       -- (MANDATORY)
            -- OCTETS:  The coded character set name of the user that
            -- submitted the job.  The method of assigning this user
            -- name will be system and/or site specific but the method
            -- must insure that the name is unique to the network that
            -- is visible to the client and target device.
            --
            -- This value SHOULD be the authenticated name of the user
            -- submitting the job.
            --
            -- In order to assist users to find their jobs for job
            -- submission protocols that don't supply a
            -- jmJobSubmissionID, the agent SHOULD maintain the jobOwner
            -- attribute for the time specified by the
            -- jmGeneralJobPersistence object, rather than the (shorter)
            -- jmGeneralAttributePersistence object.

   jobAccountName(2116),              -- OCTET STRING(SIZE(0..63))
            -- OCTETS:  Arbitrary binary information which MAY be coded
            -- character set data or encrypted data supplied by the
            -- submitting user for use by accounting services to
            -- allocate or categorize charges for services provided,
            -- such as a customer account name.
            --
            -- NOTE: This attribute NEED NOT be printable characters.
```

```
    serverAssignedJobName(2212),      -- OCTET STRING(SIZE(0..63))
        -- OCTETS:  Configuration 3 only:  The human readable string
        -- name of the job as assigned by the server that submitted
        -- the job to the device that the agent in instrumenting
        -- with this MIB.
        --
        -- NOTE - This attribute is intended for enabling a user to
        -- find his/her job that a server submitted to a device
        -- after the user submitted the job to the server when the
        -- jmJobSubmissionIDGroup is not supported by the job
        -- submission protocolimplemented.

    jobName(2313),                    -- OCTET STRING(SIZE(0..63))
        -- OCTETS:  The human readable string name of the job as
        -- assigned by the submitting user to help the user
        -- distinguish between his/her various jobs.  This name does
        -- not need to be unique.
        --
        -- This attribute is intended for enabling a user or the
        -- user's application to convey a job name that MAY be
        -- printed on a start sheet, returned in a query result, or
        -- used in notification or logging messages.
        --
        -- In order to assist users to find their jobs for job
        -- submission protocols that don't supply a
        -- jmJobSubmissionID, the agent SHOULD maintain the jobName
        -- attribute for the time specified by the
        -- jmGeneralJobPersistence object, rather than the (shorter)
        -- jmGeneralAttributePersistence object.
        --
        -- If this attribute is not specified when the job is
        -- submitted, no job name is assumed, but implementation
        -- specific defaults are allowed, such as the value of the
        -- documentName attribute of resource item of the first
        -- document in the job or the fileName attribute resource
        -- item of the first document in the job.
        --
        -- The jobName attribute is distinguished from the
        -- jobComment attribute, in that the jobName attribute is
        -- intended to permit the submitting user to distinguish
        -- between different jobs that he/she has submitted.  The
        -- jobComment attribute is intended to be free form
        -- additional information that a user might wish to use to
        -- communicate with himself/herself, such as a reminder of
        -- what to do with the results or to indicate a different
        -- set of input parameters were tried in several different
        -- job submissions.

    jobServiceTypes(2414),            -- JmJobServiceTypesTC (pg 59)
```

```
        -- INTEGER:  Specifies the type(s) of service to which the
        -- job has been submitted (print, fax, scan, etc.).  The
        -- service type is bit encoded with each job service type so
        -- that more general and arbitrary services can be created,
        -- such as services with more than one destination type, or
        -- ones with only a source or only a destination.  For
        -- example, a job service might scan, faxOut, and print a
        -- single job.  In this case, three bits would be set in the
        -- jobServiceTypes attribute, corresponding to the
        -- hexadecimal values: 0x8 + 0x20 + 0x4, respectively,
        -- yielding: 0x2C.
        --
        -- Whether this attribute is set from a job attribute
        -- supplied by the job submission client or is set by the
        -- recipient job submission server or device depends on the
        -- job submission protocol.  This attribute SHALL be
        -- implemented if the server or device has other types in
        -- addition to or instead of printing.
        --
        -- One of the purposes of this attribute is to permit a
        -- requester to filter out jobs that are not of interest.
        -- For example, a printer operator may only be interested in
        -- jobs that include printing.  That is why this attribute
        -- is in the job identification category.

jobSourceChannelIndex(25~~17~~),      -- Integer32(0..2147483647)
        -- INTEGER:  The index of the row in the associated Printer
        -- MIB[1] of the channel which is the source of the print
        -- job.
        --
        -- NOTE - the Job Monitoring MIB points to the Channel row
        -- in the Printer MIB[1], so there is no need for a port
        -- attribute in the Job Monitoring MIB, since the PWG is
        -- adding a prtChannelInformation object to the Channel
        -- table of the draft Printer MIB.

jobSourcePlatformType(26~~18~~),      -- JmJobSourcePlatformTypeTC
                                      -- (pg 26)
        -- INTEGER:  The source platform type of the immediate
        -- upstream submitter that submitted the job to the server
        -- (configuration 2) or device (configuration 1 and 3) that
        -- the agent is instrumenting.  For configuration 1, this is
        -- the type of the client that submitted the job to the
        -- device;  for configuration 2, this is the type of the
        -- client that submitted the job to the server; and for
        -- configuration 3, this is the type of the server that
        -- submitted the job to the device.

submittingServerName(27~~19~~),      -- OCTET STRING(SIZE(0..63))
```

```
                -- OCTETS:  For configuration 3 only:  The administrative
                -- name of the server that submitted the job to the device.

        submittingApplicationName(2820)  -- OCTET STRING(SIZE(0..63))
,
                -- OCTETS:  The name of the client application (not the
                -- server in configuration 3) that submitted the job to the
                -- server or device.

        jobOriginatingHost(29),          -- OCTET STRING(SIZE(0..63))
                -- OCTETS:  The name of the client host (not the server host
                -- name in configuration 3) that submitted the job to the
                -- server or device.

        deviceNameRequested(3021),       -- OCTET STRING(SIZE(0..63))
                -- OCTETS:  The administratively defined coded character set
                -- name of the target device requested by the submitting
                -- user.  For configuration 1, its value corresponds to the
                -- Printer MIB[1]: prtGeneralPrinterName object (added to
                -- the draft Printer MIB) for printers.  For configuration 2
                -- and 3, its value is the name of the logical or physical
                -- device that the user supplied to indicate to the server
                -- on which device(s) they wanted the job to be processed.

        queueNameRequested(3122),        -- OCTET STRING(SIZE(0..63))
                -- OCTETS:  The administratively defined coded character set
                -- name of the target queue requested by the submitting
                -- user.  For configuration 1, its value corresponds to the
                -- queue in the device that the agent is instrumenting.  For
                -- configuration 2 and 3, its value is the name of the queue
                -- that the user supplied to indicate to the server on which
                -- device(s) they wanted the job to be processed.
                --
                -- NOTE - typically an implementation SHOULD support either
                -- the deviceNameRequested or queueNameRequested attribute,
                -- but not both.

        physicalDeviceIndex(3223),       -- hrDeviceIndex (see HR MIB)
                                         -- AND/OR
                                         -- OCTET STRING(SIZE(0..63))
                -- INTEGER:  MULTI-ROW:  The index of the physical device
                -- MIB instance requested/used, such as the Printer MIB[1].
                -- This value is an hrDeviceIndex value.  See the Host
                -- Resources MIB[6].
                -- AND/OR
                -- OCTETS:  MULTI-ROW:  The name of the physical device to
                -- which the job is assigned.

        physicalDeviceName(24),          -- OCTET STRING(SIZE(0..63))
```

```
        -- OCTETS:  MULTI ROW:  The name of the physical device to
        -- which the job is assigned.

    numberOfDocuments(3325),         -- Integer32(0..2147483647)
        -- INTEGER:  The number of documents in this job.  If this
        -- attribute is not present, the number of documents SHALL
        -- be 1.

    fileName(3426),                  -- OCTET STRING(SIZE(0..63))
        -- OCTETS:  MULTI-ROW:  The coded character set file name of
        -- the document.
        --
        -- There is no restriction on the same file name in multiple
        -- rows.

    documentName(3527),              -- OCTET STRING(SIZE(0..63))
        -- OCTETS:  MULTI-ROW:  The coded character set name of the
        -- document.
        --
        -- There is no restriction on the same document name in
        -- multiple rows.

    jobComment(3628),                -- OCTET STRING(SIZE(0..63))
        -- OCTETS:  An arbitrary human-readable coded character text
        -- string supplied by the submitting user or the job
        -- submitting application program for any purpose.  For
        -- example, a user might indicate what he/she is going to do
        -- with the printed output or the job submitting application
        -- program might indicate how the document was produced.
        --
        -- The jobComment attribute is not intended to be a name;
        -- see the jobName attribute.

    documentFormatIndex(3729),       -- Integer32(0..2147483647)
        -- INTEGER:  MULTI-ROW:  The index in the
        -- prtInterpreterTable interpreter language family index in
        -- the Printer MIB[1] of the page description language (PDL)
        -- or control language interpreter prtInterpreterLangFamily
        -- object, that this job requires/uses.  A document or a job
        -- MAY use more than one PDL or control language.
        --
        -- NOTE - As with all intensive attribute items where
        -- multiple rows are allowed, there SHALL be only one
        -- distinct row for each distinct interpreterPDL; there
        -- SHALL be no duplicates.
        --
        -- NOTE - This attribute type is intended to be used with an
        -- agent that implements the Printer MIB and SHALL not be
        -- used if the agent does not implement the Printer MIB.
```

```
                -- Such ans agent SHALL use the documentFormatType attribute
                   instead.

        documentFormatType(3830),             -- PrtInterpreterLangFamilyTC
                                              -- AND/OR
                                              -- OCTET STRING(SIZE(0..63))
           -- INTEGER:  MULTI-ROW:  The interpreter language family
           -- corresponding to the Printer MIB[1]
           -- prtInterpreterLangFamily object, that this job
           -- requires/uses.  A document or a job MAY use more than one
           -- PDL or control language.
           --
           -- NOTE - This attribute is represented by a type 2 enum
           -- defined in the draft Printer MIB[1], but is not in RFC
           -- 1759.
           --
           -- AND/OR
           --
           -- OCTETS:  MULTI-ROW:  The document format registered as a
           -- MIME type, i.e., the name of the MIME type.
           --
           -- NOTE - IPP[3] uses MIME type keywords to identify
           -- document formats.


        -- ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
        -- Job Parameter attributes
        --
        -- The following attributes represent input parameters
        -- supplied by the submitting client in the job submission
        -- protocol.
        -- ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

        jobPriority(5031),                    -- Integer32(1..100)
           -- INTEGER:  The priority for scheduling the job. It is used
           -- by servers and devices that employ a priority-based
           -- scheduling algorithm.
           --
           -- A higher value specifies a higher priority. The value 1
           -- is defined to indicate the lowest possible priority (a
           -- job which a priority-based scheduling algorithm SHALL
           -- pass over in favor of higher priority jobs). The value
           -- 100 is defined to indicate the highest possible priority.
           -- Priority is expected to be evenly or 'normally'
           -- distributed across this range. The mapping of vendor-
           -- defined priority over this range is implementation-
           -- specific.

        jobProcessAfterDateAndTime(5132  -- DateAndTime (SNMPv2-TC)
```

```
    ),
        -- INTEGER:  The calendar date and time of day after which
        -- the job SHALL become a candidate to be scheduled for
        -- processing.  If the value of this attribute is in the
        -- future, the server SHALL set the value of the job's
        -- jmJobState/jobState object/attribute to pendingHheld and
        -- add the jobProcessAfterSpecified bit value to the job's
        -- jmJobStateReasons1 objectattribute and SHALL not schedule
        -- the job for processing until the specified date and time
        -- has passed.  When the specified date and time arrives,
        -- the server SHALL remove the jobProcessAfterSpecified bit
        -- value from the job's jmJobStateReasons1 objectattribute
        -- and, if no other reasons remain, SHALL change the job's
        -- jmJobState objectand the job's jobState attribute to
        -- pending so that the job becomes a candidate for being
        -- scheduled on devices(s).
        --
        -- The agent SHALL assign an empty value to the
        -- jobProcessAfterDateAndTime attribute when no process
        -- after time has been specified, so that the job SHALL be a
        -- candidate for processing immediately.

    jobHold(52),                       -- Integer32(0..1)
        -- INTEGER:  If the value is 1, a client has explicitly
        -- specified that the job is to be held until explicitly
        -- released.  Until the job is explicitly released by a
        -- client, the job SHALL be in the pendingHeld state with
        -- the jobHoldSpecified value in the jmJobStateReasons1
        -- attribute.

    jobHoldUntil(5333),                -- OCTET STRING(SIZE(0..63))
        -- OCTETS:  The named time period during which the job SHALL
        -- become a candidate for processing, such as 'no-hold',
        -- 'evening', 'night', 'weekend', 'second-shift', 'third-
        -- shift', etc., as defined by the system administrator.
        -- Until that time period arrives, the job SHALL be in the
        -- pendingHheld state with the jobHoldUntilSpecified value
        -- in the jmJobStateReasons1 objectattribute.

    outputBinIndex(5434),              -- Integer32(0..2147483647)
                                       -- AND/OR
                                       -- OCTET STRING(SIZE(0..63))
        -- INTEGER:  MULTI-ROW:  The output subunit index in the
        -- Printer MIB[1] of the output bin to which all or part of
        -- the job is placed in.
        -- AND/OR
        -- OCTETS:  the name of the output bin to which all or part
        -- of the job is placed in.
```

```
   outputBinName(35),                  -- OCTET STRING(SIZE(0..63))
       -- OCTETS:  MULTI ROW:  The name of the output bin to which
       -- all or part of the job is placed in.

   sides(5536),                        -- Integer32(-2..1)
       -- INTEGER:  MULTI-ROW:  The number of sides that any
       -- document in this job requires/used.

   finishing(5637),                    -- JmFinishingTC (pg 27)
       -- INTEGER:  MULTI-ROW:  Type of finishing that any document
       -- in this job requires/used.


   -- +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   -- Image Quality attributes (requested and consumed)
   --
   -- For devices that can vary the image quality.
   -- +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

   printQualityRequested(7038),    -- JmPrintQualityTC (pg 28)
       -- INTEGER:  MULTI-ROW:  The print quality selection
       -- requested for document in the job for printers that allow
       -- quality differentiation.

   printQualityUsed(7139),         -- JmPrintQualityTC (pg 28)
       -- INTEGER:  MULTI-ROW:  The print quality selection
       -- actually used by documents in the job for printers that
       -- allow quality differentiation.

   printerResolutionRequested(72),  -- JmPrinterResolutionTC
                                    -- (pg 29)
       -- INTEGER:  MULTI-ROW:  The print quality selection
       -- requested for document in the job for printers that allow
       -- quality differentiation.

   printerResolutionUsed(73),       -- JmPrinterResolutionTC
                                    -- (pg 29)
       -- INTEGER:  MULTI-ROW:  The print quality selection
       -- actually used by documents in the job for printers that
       -- allow quality differentiation.

   tonerEcomonyRequested(7440),     -- JmTonerEconomyTC (pg 30)
       -- INTEGER:  MULTI-ROW:  The print quality selection
       -- requested for documents in the job for printers that
       -- allow toner quality differentiation.

   tonerEcomonyUsed(7541),          -- JmTonerEconomyTC (pg 30)
       -- INTEGER:  MULTI-ROW:  The print quality selection
       -- actually used by documents in the job for printers that
```

```
                -- allow toner quality differentiation.

    tonerDensityRequested(7642),       -- Integer32(1..20)
        -- INTEGER:  MULTI-ROW:  The toner density requested for
        -- documents in this job for devices that can vary toner
        -- density levels.  Level 1 is the lowest density and level
        -- 20 is the highest density level.  Devices with a smaller
        -- range, SHALL map the 1-20 range evenly onto the
        -- implemented range.

    tonerDensityUsed(7743),            -- Integer32(1..20)
        -- INTEGER:  MULTI-ROW:  The toner density used by documents
        -- in this job for devices that can vary toner density
        -- levels.  Level 1 is the lowest density and level 20 is
        -- the highest density level.  Devices with a smaller range,
        -- SHALL map the 1-20 range evenly onto the implemented
        -- range.


    -- +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
    -- Job Progress attributes (requested and consumed)
    --
    -- Pairs of these attributes can be used by monitoring
    -- applications to show 'thermometers' of progress to users.
    -- +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

    jobCopiesRequested(9044),          -- Integer32(-2..2147483647)
        -- INTEGER:  The number of copies of the entire job that are
        -- to be produced.

    jobCopiesCompleted(9145),          -- Integer32(-2..2147483647)
        -- INTEGER:  The number of copies of the entire job that
        -- have been completed so far.

    documentCopiesRequested(9246),     -- Integer32(-2..2147483647)
        -- INTEGER:  The total count of the number of document
        -- copies requested.  If there are documents A, B, and C,
        -- and document B is specified to produce 4 copies, the
        -- number of document copies requested is 6 for the job.
        --
        -- This attribute SHALL be used only when a job has multiple
        -- documents.  The jobCopiesRequested attribute SHALL be
        -- used when the job has only one document.

    documentCopiesCompleted(9347),     -- Integer32(-2..2147483647)
        -- INTEGER:  The total count of the number of document
        -- copies completed so far for the job as a whole.  If there
        -- are documents A, B, and C, and document B is specified to
        -- produce 4 copies, the number of document copies starts a
```

```
      -- 0 and runs up to 6 for the job as the job processes.
      --
      -- This attribute SHALL be used only when a job has multiple
      -- documents.  The jobCopiesCompleted attribute SHALL be
      -- used when the job has only one document.

   jobKOctetsRequested(48),           -- Integer32(-2..2147483647)
      -- INTEGER:  The total number of K (1024) octets being
      -- requested to be processed in the job, including document
      -- and job copies.  The agent shall round the actual number
      -- of octets up to the next highest K.  Thus 0 octets shall
      -- be represented as 0, 1 1024 octets shall be represented
      -- as 1, 1025 2048 shall be represented as 2, etc.
      --
      -- The server/device may update the value of this attribute
      -- after each document has been transferred to the
      -- server/device or the server/device may provide this value
      -- after all documents have been transferred to the
      -- server/device, depending on implementation.  In other
      -- words, while the job is in the held state with the
      -- jobStateReasons1 attribute containing a documentsNeeded
      -- or preProcessing value, the value of the
      -- jobKOctetsRequested attribute depends on implementation
      -- and may not correctly reflect the size of the job.
      --
      -- In computing this value, the server/device shall include
      -- the multiplicative factors contributed by (1) the number
      -- of document copies, and (2) the number of job copies,
      -- independent of whether the device can process multiple
      -- copies of the job or document without making multiple
      -- passes over the job or document data and independent of
      -- whether the output is collated or not.  Thus the
      -- server/device computation is independent of the
      -- implementation and shall be:
      --
      --     (1) Document contribution:  Multiply the size of each
      --     document in octets by the number of document copies
      --     of that document.
      --
      --     (2) Add each document contribution together.
      --
      --     (3) Job copy contribution:  Multiply the job size by
      --     the number of job copies.
      --
      --     (4) Round up the result to the next higher K (1024
      --     multiple).

   jobKOctetsTransferred(9449),     -- Integer32(-2..2147483647)
      -- INTEGER:  The number of K (1024) octets transferred to
```

```
     -- the server or device that the agent is instrumenting.
     -- This count is independent of the number of copies of the
     -- job or documents that will be produced, but is just a
     -- measure of the number of bytes transferred to the server
     -- or device.
     --
     -- The agent SHALL round the actual number of octets
     -- transferred~~completed~~ up to the next higher K.  Thus 0
     -- octets SHALL be~~is~~ represented as 0, 1-1024~~3~~ octets~~,~~ SHALL
     -- BE~~is~~ represented as 1, 1025~~4~~-2048~~7~~ SHALL be~~is~~ 2, etc.
     -- When the job completes, the values of the
     -- jmJ~~ob~~JobKOctetsRequested object and the
     -- jobKOctetsTransferred attribute~~s~~ SHALL be equal.
     --
     -- NOTE - The jobKOctetsTransferred can be used in the
     -- numerator with the jmJ~~ob~~JobKOctetsRequested object~~attribute~~
     -- in the denominator in order to produce a "thermometer"
     -- that indicates the progress of the job for agents that do
        not implement~~instrument~~ the
        jmJ~~ob~~JobKOctetsProcessed~~Completed~~ object~~attribute~~.
```

```
   ~~jobKOctetsCompleted(50),        -- Integer32(-2..2147483647)~~
     ~~-- INTEGER:  The number of K (1024) octets currently~~
     ~~-- processed by the server or device, including document and~~
     ~~-- job copies.  For printing, the completed count only~~
     ~~-- includes processing (interpreting) if the implementation~~
     ~~-- distinguishes between the processing and printing states;~~
     ~~-- otherwise, the completed count includes both processing~~
     ~~-- (interpreting) and marking combined together.  For~~
     ~~-- scanning, the completed count only includes scanning, if~~
     ~~-- the implementation distinguishes between the processing~~
     ~~-- and (to be registered) scanning states; otherwise the~~
     ~~-- completed count includes both scanning and processing~~
     ~~-- (formatting).~~
     ~~--~~
     ~~-- The agent shall round the actual number of octets~~
     ~~-- completed up to the next higher K.  Thus 0 octets is~~
     ~~-- represented as 0, 1-1023, is represented as 1, 1024-2047~~
     ~~-- is 2, etc.  When the job completes, the values of the~~
     ~~-- jobKOctetsRequested and the jobKOctetsCompleted~~
     ~~-- attributes shall be equal.~~
     ~~--~~
     ~~-- For multiple copies generated from a single data stream,~~
     ~~-- the value shall be incremented as if each copy was~~
     ~~-- printed from a new data stream without resetting the~~
     ~~-- count between copies.  See the pagesCompletedCurrentCopy~~
     ~~-- attribute that is reset on each document copy.~~
     ~~--~~
     ~~-- NOTE - The jobKOctetsCompleted can be used in the~~
```

```
        -- numerator with the jobKOctetsRequested attribute in the
        -- denominator in order to produce a "thermometer" that
        -- indicates the progress of the job.


    -- ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
    -- Impression attributes
    --
    -- For a print job, an impression is the marking of the
    -- entire side of a sheet.  Two-sided processing involves two
    -- impressions per sheet.  Two-up is the placement of two
    -- logical pages on one side of a sheet and so is still a
    -- single impression.  See also jmJobImpressionsRequested and
    -- jmJobImpressionsCompleted objects in the jmJobTable on page
    -- 85.
    -- ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

    impressionsSpooled(11051),        -- Integer32(-2..2147483647)
        -- INTEGER:  The number of impressions spooled to the server
        -- or device for the job so far.

    impressionsSentToDevice(11152),  -- Integer32(-2..2147483647)
        -- INTEGER:  The number of impressions sent to the device
        -- for the job so far.

    impressionsInterpreted(11253),    -- Integer32(-2..2147483647)
        -- INTEGER:  The number of impressions interpreted for the
        -- job so far.

    impressionsRequested(54),          -- Integer32(-2..2147483647)
        -- INTEGER:  The number of impressions requested by this job
        -- to produce.

    impressionsCompleted(55),          -- Integer32(-2..2147483647)
        -- INTEGER:  The total number of impressions completed by
        -- the device for this job so far.  For printing, the
        -- impressions completed includes interpreting, marking, and
        -- stacking the output.  For other types of job services,
        -- the number of impressions completed includes the number
        -- of impressions processed.

    impressionsCompletedCurrentCopy(11356  -- Integer32(-2..
    ),                                      -- 2147483647)
        -- INTEGER:  The number of impressions completed by the
        -- device for the current copy of the current document so
        -- far.  For printing, the impressions completed includes
        -- interpreting, marking, and stacking the output.  For
        -- other types of job services, the number of impressions
        -- completed includes the number of impressions processed.
```

```
      --
      -- This value SHALL be reset to 0 for each document in the
      -- job and for each document copy.

  fullColorImpressionsCompleted(114),   -- Integer32(-2..
                                        -- 2147483647)
      -- INTEGER:  The number of full color impressions completed
      -- by the device for this job so far.  For printing, the
      -- impressions completed includes interpreting, marking, and
      -- stacking the output.  For other types of job services,
      -- the number of impressions completed includes the number
      -- of impressions processed. Full color impressions are
      -- typically defined as those requiring 3 or more colorants,
      -- but this MAY vary by implementation.
      --
  highlightColorImpressionsCompleted(115),  --  Integer32(-2..
                                            -- 2147483647)
      -- INTEGER:  The number of highlight color impressions
      -- completed by the device for this job so far.  For
      -- printing, the impressions completed includes
      -- interpreting, marking, and stacking the output.  For
      -- other types of job services, the number of impressions
      -- completed includes the number of impressions processed.
      -- Highlight color impressions are typically defined as
      -- those requiring black plus one other colorant, but this
      -- MAY vary by implementation.


  -- +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
  -- Page attributes
  --
  -- A page is a logical page.  Number up can impose more than
  -- one page on a single side of a sheet.  Two-up is the
  -- placement of two logical pages on one side of a sheet so
  -- that each side counts as two pages.
  -- +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

  pagesRequested(13057),            -- Integer32(-2..2147483647)
      -- INTEGER:  The number of logical pages requested by the
      -- job to be processed.

  pagesCompleted(13158),            -- Integer32(-2..2147483647)
      -- INTEGER:  The total number of logical pages completed for
      -- this job so far.

  pagesCompletedCurrentCopy(13259  -- Integer32(-2..2147483647)
  ),
      -- INTEGER:  The number of logical pages completed for the
      -- current copy of the document so far.  This value SHALL be
```

```
         -- reset to 0 for each document in the job and for each
         -- document copy.


    -- +++++++++++++++++++++++++++++++++++++++++++++++++++++++++
    -- Sheet attributes
    --
    -- The sheet is a single piece of a medium, whether printing
    -- on one or both sides.
    -- +++++++++++++++++++++++++++++++++++++++++++++++++++++++++

    sheetsRequested(15060),          -- Integer32(-2..2147483647)
         -- INTEGER:  The total number of medium sheets requested to
         -- be processed for this job.

    sheetsCompleted(15161),          -- Integer32(-2..2147483647)
         -- INTEGER:  The total number of medium sheets that have
         -- completed marking and stacking for the entire job so far
         -- whether those sheets have been processed on one side or
         -- on both.

    sheetsCompletedCurrentCopy(15262  -- Integer32(-2..2147483647)
    ),
         -- INTEGER:  The number of medium sheets that have completed
         -- marking and stacking for the current copy of a document
         -- in the job so far whether those sheets have been
         -- processed on one side or on both.
         --
         -- The value of this attribute SHALL be reset to 0 as each
         -- document in the job starts being processed and for each
         -- document copy as it starts being processed.


    -- +++++++++++++++++++++++++++++++++++++++++++++++++++++++++
    -- Resources attributes (requested and consumed)
    --
    -- Pairs of these attributes can be used by monitoring
    -- applications to show 'thermometers' of usage to users.
    -- +++++++++++++++++++++++++++++++++++++++++++++++++++++++++

    mediumRequestedType(17063),       -- JmMediumTypeTC (pg 30)
                                      -- AND/OR
                                      -- OCTET STRING(SIZE(0..63))
         -- INTEGEROCTETS:  MULTI-ROW:  The type of the medium that
         -- is required by the job.
         -- AND/OR
         -- OCTETS:  the name of the medium that is required by the
            job.
```

```
mediumRequestedName(64),        -- OCTET STRING(SIZE(0..63))
    -- OCTETS:  MULTI-ROW:  The name of the medium that is
    -- required by the job.

mediumConsumedName(17165),        -- OCTET STRING(SIZE(0..63))
                                  -- AND
                                  -- Integer32(-2..2147483647)
    -- OCTETS: MULTI-ROW:  The name of the medium
    -- AND
    -- INTEGER:  the number of sheets that have been consumed so
    -- far whether those sheets have been processed on one side
    -- or on both.  This attribute SHALL have both values.

colorantRequestedIndex(17266),    -- Integer32(0..2147483647)
                                  -- AND/OR
                                  -- OCTET STRING(SIZE(0..63))
    -- INTEGER:  MULTI-ROW:  The index (prtMarkerColorantIndex)
    -- in the Printer MIB[1] of the colorant requested.
    -- AND/OR
    -- OCTETS:   the name of the colorant requested.

colorantRequestedName(67),        -- OCTET STRING(SIZE(0..63))
    -- OCTETS:  MULTI-ROW:  The name of the colorant requested.

colorantConsumedIndex(17368),     -- Integer32(0..2147483647)
                                  -- AND/OR
                                  -- OCTET STRING(SIZE(0..63))
    -- INTEGER:  MULTI-ROW:  The index (prtMarkerColorantIndex)
    -- in the Printer MIB[1] of the colorant consumed.
    -- AND/OR
    -- OCTETS: the name of the colorant consumed.

colorantConsumedName(69),         -- OCTET STRING(SIZE(0..63))
    -- OCTETS:  MULTI-ROW:  The name of the colorant consumed.



    -- +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
-- Time attributes (set by server or device)
--
-- This section of attributes are ones that are set by the
-- server or device that accepts jobs.  Two forms of time are
-- provided.  Each form is represented in a separate attribute.
-- See section 4.2 on page 18 and section 4.3 on page 19 for the
-- conformance requirements for agents and monitoring
-- applications, respectively.  The two forms are:
--
-- DateAndTime is an 8 or 11 octet binary encoded year,
-- month, day, hour, minute, second, deci-second with
-- optional offset from UTC.  See SNMPv2-TC.
```

```
--
-- NOTE: DateAndTime is not printable characters; it is
-- binary.
--
-- JmTimeStampTC is the time of day measured in the number of
-- seconds since the system was booted.  See page 26.
-- ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

jobSubmissionToServerDateAndTim  -- JmTimeStampTC (pg 26)
e(19070),                        -- AND/OR
                                 -- DateAndTime (SNMPv2-TC)
    -- INTEGER:  Configuration 2 and 3:  The time
    -- AND/OR
    -- OCTETS:  tConfiguration 2 and 3:  The date and time that
    -- the job was submitted to the server.

jobSubmissionToDeviceDateAndTim  -- JmTimeStampTC (pg 26)
e(19171),                        -- AND/OR
                                 -- DateAndTime (SNMPv2-TC)
    -- INTEGER:  Configuration 1 and 3:  The time
    -- AND/OR
    -- OCTETS:  tConfiguration 1 and 3:  The date and time that
    -- the job was submitted to the device.

timeSinceJobWasSubmittedToDevice(192),   -- Integer32(0..
                                         -- 2147483647)
    -- INTEGER:  The time in seconds since the job was submitted
    -- to the device.

jobSubmissionToDeviceTimeStamp(72),   -- JmTimeStampTC (pg )
    -- INTEGER:  The time that the job was submitted.

jobStartedBeingHeldTimeStamp(19373  -- JmTimeStampTC (pg 26)
),
    -- INTEGER:  The time that the job started being held, i.e.,
    -- the time that the job entered the pendingHheld state most
    -- recently.  If the job has never entered the pendingHheld
    -- state, then the value SHALL be 0 or the attribute SHALL
    -- not be present in the table.

jobStartedProcessingDateAndTime  -- JmTimeStampTC (pg 26)
(19474),                         -- AND/OR
                                 -- DateAndTime (SNMPv2-TC)
    -- INTEGER:  The time
    -- AND/OR
    -- OCTETS:  tThe date and time that the job started
    -- processing.

timeSinceStartedProcessing(195),  -- Integer32(-2..2147483647)
```

```
                  -- INTEGER:  The time in milliseconds since the job started
                  -- processing.

              jobStartedProcessingTimeStamp(75),  -- JmTimeStampTC (pg )
                  -- INTEGER:  The time that the job started processing.

              jobCompletedDateAndTime(19676),  -- JmTimeStampTC (pg 26)
                                               -- AND/OR
                                               -- DateAndTime (SNMPv2-TC)
                  -- INTEGER:   The time
                  -- AND/OR
                  -- OCTETS:   tThe date and time that the job completed
                  -- processing and the medium is completely stacked in the
                  -- output bin, i.e., when the job entered the completed,
                  -- canceled, or aborted -state.

              jobCompletedTimeStamp(77),       -- JmTimeStampTC (pg )
                  -- INTEGER:  The time that the job completed processing and
                  -- the medium is completely stacked in the output bin, i.e.,
                  -- when the job entered the completed state.

              timeSinceCompleted(197),         -- Integer32(-2..2147483647)
                  -- INTEGER:  The time in milliseconds since the job
                  -- completed processing and the medium was completely
                  -- stacked in the output bin, i.e., since the job entered
                  -- the completed, canceled, or aborted state.

              jobProcessingCPUTime(19878)      -- Integer32(-2..2147483647)
                  -- INTEGER:  The amount of CPU time that the job has been
                  -- processing in seconds, i.e., in the processing job state.
                  -- If the device stops and/or the job enters the
                  -- processingStopped stateneeds attention, that elapsed time
                  -- SHALL not be included.  In other words, the
                  -- jobProcessingCPUTime value SHOULD be relatively
                  -- repeatable when the same job is submitted again.

1144      }
1145
1146
1147

1148

1149   JmJobServiceTypesTC ::= TEXTUAL-CONVENTION
1150       STATUS      current
1151       DESCRIPTION
1152          "Specifies the type(s) of service to which the job has been
1153          submitted (print, fax, scan, etc.).  The service type is
1154          represented as an enum that is bit encoded with each job service
1155          type so that more general and arbitrary services can be created,
```

1156          such as services with more than one destination type, or ones
1157          with only a source or only a destination.  For example, a job
1158          service might **scan**, **faxOut**, and **print** a single job.  In this
1159          case, three bits would be set in the **jobServiceTypes** attribute,
1160          corresponding to the hexadecimal values: **0x8** + **0x20** + 0**x4**,
1161          respectively, yielding: **0x2C**.
1162
1163          Whether this attribute is set from a job attribute supplied by
1164          the job submission client or is set by the recipient job
1165          submission server or device depends on the job submission
1166          protocol.  With either implementation, the agent SHALL return a
1167          non-zero value for this attribute indicating the type of the
1168          job.
1169
1170          One of the purposes of this attribute is to permit a requester
1171          to filter out jobs that are not of interest.  For example, a
1172          printer operator MAY only be interested in jobs that include
1173          printing.  That is why the attribute is in the job
1174          identification category.
1175
1176          The following service component types are defined (in
1177          hexadecimal) and are assigned a separate bit value for use with
1178          the **jobServiceTypes** attribute:
1179
1180     **other**                                      0x1
1181          The job contains some document production instructions that are
1182          not one of the identified types.
1183
1184     **unknown**                                    0x2
1185          The job contains some document production instructions whose
1186          type is unknown to the agent.
1187
1188     **print**                                      0x4
1189          The job contains some document production instructions that
1190          specify printing
1191
1192     **scan**                                       0x8
1193          The job contains some document production instructions that
1194          specify scanning
1195
1196     **faxIn**                                      0x10
1197          The job contains some document production instructions that
1198          specify receive fax
1199
1200     **faxOut**                                     0x20
1201          The job contains some document production instructions that
1202          specify sending fax
1203
1204     **getFile**                                    0x40

```
1205          The job contains some document production instructions that
1206          specify accessing files or documents
1207
1208      putFile                                 0x80
1209          The job contains some document production instructions that
1210          specify storing files or documents
1211
1212      mailList                                0x100
1213          The job contains some document production instructions that
1214          specify distribution of documents using an electronic mail
1215          system.
1216
1217
1218      These bit definitions are the equivalent of a type 2 enum except
1219      that combinations of them MAY be used together.  See section 7.1.2
1220      on page 21."
1221
1222      SYNTAX        INTEGER(0..2147483647)    -- 31 bits, all but sign bit
1223
1224
1225
1226
1227  JmJobStateReasons1TC ::= TEXTUAL-CONVENTION
1228      STATUS        current
1229      DESCRIPTION
1230          "This textual-convention is used with the jmJobStateReasons1
1231          objectattribute to provides additional information regarding the
1232          jmJobState/jobState object values/attribute.  The
1233          jobStateReasons1 attributes identifies the reason or reasons
1234          that the job is in the held, pending, processing, printing,
1235          needsAttention, canceled, or completed state.  The server shall
1236          indicate the particular reason(s) by setting the value of the
1237          jobStateReasons1 attribute.  While the job states cannot be
1238          added to without impacting deployed clients, it is the intent
1239          that additional JmJobStateReasons1TC enums can be defined
1240          without impacting deployed clients.  In other words, the
1241          JmJobStateReasons1TC is intended to be extensible.
1242
1243          When the job does not have any reasons for being in its current
1244          state, the server shall set the value of the jobStateReasons1
1245          attribute to zeros.
1246
1247          Companion job state reasons TCs: JmJobStateReasons2TC,
1248          JmJobStateReasons3TC, JmJobStateReasons4TC, are defined/reserved
1249          for additional 31*3 = 93 job state reasons.  This is a type 2
1250          bit definition.  See section  on page .
```

The following standard values are defined (in hexadecimal) as
*powers of two,* since multiple values MAY be used at the same
time.

NOTE - The Job Monitoring MIB contains a superset of the IPP
values[3] for the IPP 'job-state-reasons' attribute, since the
Job Monitoring MIB is intended to cover other job submission
protocols as well.  Also some of the names of the reasons have
been changed from 'printer' to 'device', since the Job
Monitoring MIB is intended to cover additional types of devices,
including input devices, such as scanners.÷

NOTE - For easy of understanding the order of the reasons is
presented in the order in which the reason is most likely to
occur.

**other**                                    **0x1**
    The job state reason is not one of the standardized or
    registered reasons.

**unknown**                                  **0x2**
    The job state reason is not known to the agent or is
    indeterminent.

**jobIncoming**~~documentsNeeded~~           **0x4**
    The job has been accepted by the server or device, but the
    server or device is ~~in the held state because the server or~~
    ~~device is~~expected ~~waiting for~~ (1) additional operations to
    finish creating the job and/or (2) is accessing/accepting
    document data~~the job's files to start and/or finish being~~
    ~~transferred before the job can be scheduled to be processed~~.

**jobOutgoing**                              **0x8**
    Configuration 2 only:  The server is transmitting the job to the
    device.

**jobHoldSpecified**~~Set~~                  **0x10**~~8~~
    The ~~job is in the **held** state because the client specified that~~
    ~~the job is to be held~~value of the job's **jobHold(52)** attribute
    (see page 48) is **TRUE**, either set when the job was created or
    subsequently by an explicit modify job operation.  The job SHALL
    NOT be a candidate for processing until this reason is removed
    and there are no other reasons to hold the job.

**jobHoldUntilSpecified**                    **0x20**~~40000~~
    The value of the job's **jobHoldUntil(**53~~33~~**)** (see page 48)
    attribute specifies a time period ~~was specified for a named time~~
    ~~period~~that is still in the future, either set when the job was

```
1300        created or subsequently by an explicit modify job operation.
1301        The job SHALL NOT be a candidate for processing until this
1302        reason is removed and there are no other reasons to hold the
1303        job.  The job remains in the held state until the time period
1304        arrives and there are no other reasons to hold the job.
1305
1306    jobProcessAfterSpecified              0x4010
1307        The value of the job's jobProcessAfterDateAndTime(5132) (see
1308        page 48) attribute specifies a timejob is in the held state
1309        because the client specified a time specification reflected in
1310        the value of the job's  attribute that is still in the future,
1311        either set when the job was created or subsequently by an
1312        explicit modify job operation.  The job SHALL NOT be a candidate
1313        for processing until this reason is removed and there are no
1314        other reasons to hold the job.
1315
1316    requiredRresourcesAreNotNotReady      0x8020
1317        The job is in the held state because Aat least one of the
1318        resources needed by the job, such as media, fonts, resource
1319        objects, etc., is not ready on any of the physical devices for
1320        which the job is a candidate. This condition MAY be detected
1321        when the job is accepted, or subsequently while the job is
1322        pending or processing, depending on implementation.
1323
1324
1325    deviceStoppedPartly                   0x100
1326        One or more, but not all, of the devices to which the job is
1327        assigned are stopped.  If all of the devices are stopped (or the
1328        only device is stopped), the deviceStopped reason SHALL be used.
1329
1330    deviceStopped                         0x200
1331        The device(s) to which the job is assigned is (are all) stopped.
1332
1333    jobPrinting                           0x400
1334        The output device is marking media. This attribute is useful for
1335        servers and output devices which spend a great deal of time
1336        processing when no marking is happening and then want to show
1337        that marking is now happening.
1338
1339    jobCeanceledByUser                    0x800200
1340        The job is in the canceled,  state having wasbeen canceled by
1341        the user, i.e., by a user whose name is the same as the value of
1342        the job's jobOwner attribute.
1343
1344    jobCeanceledByOperator                0x1000400
1345        The job wasis in the canceled state having been canceled by the
1346        operator, i.e., by a user whose name is different than the value
1347        of the job's jobOwner attribute.
1348
```

1349    **abortedBySystem**                               0x2000800
1350        The job wasis in the **canceled**, state having been aborted by the
1351        system.  NOTE - this reason is needed only when the job is not
1352        placed in the **aborted** job state.
1353
1354
1355    **jobCompletedSsuccessfullyCompletion**    0x400040
1356        The job is in the **completed** state havingcompleted successfully.
1357
1358    **jobCeompletedWithWarnings**                    0x800080
1359        The job is in the **canceled** or **completed** states having completed
1360        with warnings.
1361
1362    **jobCeompletedWithErrors**                      0x10000100
1363        The job is in the **canceled** or **completed** states having completed
1364        with errors (and possibly warnings too).
1365
1366    The following additional job state reasons have been added to
1367    specify sub states of the **held** or **completed** states that may be used
1368    to represent job states that are in ISO DPA[2] and other job
1369    submission protocols:
1370
1371    jobPreProcessing                          0x4000The job has been created
1372        on the server or device but the submitting client is in the
1373        process of adding additional job components and no documents
1374        have started processing.  The job maybe in the process of being
1375        checked by the server/device for attributes, defaults being
1376        applied, a device being selected, etc.
1377
1378    **jobPaused**                                     0x200008000
1379        The job has been indefinitely suspended by a client issuing an
1380        operation to suspend the job so that other jobs may proceed
1381        using the same devices.  The client MAY issue an operation to
1382        resume the paused job at any time, in which case the agent SHALL
1383        remove the **jobPaused** values from the job's **jmJobStateReasons1**
1384        object and the server or device places the job in the held or
1385        pending states and the job is eventually resumed at or near the
1386        point where the job was paused.
1387
1388    **jobInterrupted**                               0x4000010000
1389        The job has been interrupted while processing by a client
1390        issuing an operation that specifies another job to be run
1391        instead of the current job.  The server or device will
1392        automatically resume the interrupted job when the interrupting
1393        job completes.
1394
1395    **jobRetained**                                  0x8000020000
1396        The job is being retained by the server or device with all of
1397        the job's document data (and submitted resources, such as fonts,

1398     logos, and forms, if any).  Thus a client could issue an
1399     ~~operation to resubmit the job (or a copy of the job). after~~
1400     ~~processing and all of the media have been successfully stacked~~
1401     ~~in the output bin(s).~~
1402
1403     ~~The job (1) has completed successfully or with warnings or~~
1404     ~~errors, (2) has been aborted while printing by the~~
1405     ~~server/device, or (3) has been canceled by the submitting user~~
1406     ~~or operator before or during processing.  The job's~~
1407     ~~jobStateReasons1 attribute shall contain the reasons that the~~
1408     ~~job has entered the completed state.~~
1409
1410     ~~While the jobRetained state reason is , all of the job's~~
1411     ~~document data (and submitted resources, such as fonts, logos,~~
1412     ~~and forms, if any) are retained by the server or device; thus a~~
1413     ~~client could issue an operation to resubmit the job (or a copy~~
1414     ~~of the job).~~  When a client could no longer resubmit the job,
1415     such as after the document data has been discarded, the agent
1416     SHALL remove the **jobRetained** value from the **jmJobStateReasons1**
1417     object.
1418
1419     These bit definitions are the equivalent of a type 2 enum except
1420     that combinations of bits~~them~~ may be used together.  See section
1421     7.1.2 on page 21.  The remaining bits are reserved for future
1422     standardization and/or registration."
1423
1424     SYNTAX        **INTEGER(0..2147483647)**   **--** 31 bits, all but sign bit
1425
1426
1427
1428
1429
1430     **JmJobStateReasons2TC** ::= TEXTUAL-CONVENTION
1431         STATUS      current
1432         DESCRIPTION
1433             "This textual-convention is used with the **jobStateReasons2**
1434             attribute to provides additional information regarding the
1435             **jmJobState**~~/jobState~~ object~~/attribute~~.  See the description under
1436             **JmJobStateReasons1TC** on page 60.
1437
1438             The following standard values are defined (in hexadecimal) as
1439             *powers of two,* since multiple values may be used at the same
1440             time:
1441
1442     **cascaded**                                  **0x1**
1443         A~~fter n~~the outbound gateway has transmitted ~~retrieves~~ all of the
1444         job's ~~job~~ and document attributes and data to another spooling
1445         system.~~, it stores the information into a spool directory.  Once~~
1446         ~~it has done this, it sends the supervisor a job processing event~~

1447     ~~with this **job-state-reason** which tells the supervisor to~~
1448     ~~transition to a new job state.~~
1449
1450     **deletedByAdministrator**        **0x2**
1451     The administrator has deleted the job.~~issued a **Delete** operation~~
1452     ~~on the job or a **Clean** operation on the server or queue~~
1453     ~~containing the job; therefore the job MAY have been canceled~~
1454     ~~before or during processing, and will have no retention period~~
1455     ~~or completion period.~~
1456
1457     **discardTimeArrived**        **0x4**
1458     The job has been deleted ~~(**canceled** with the **job-retention-period**~~
1459     ~~set to **0**)~~ due to the fact that the time specified by the job's
1460     **job-discard-time** has arrived ~~[if the job had already completed,~~
1461     ~~the only action that would have occurred is that the **job-**~~
1462     ~~**retention-period** would be set to **0** and the job is deleted]~~.
1463
1464     **postProcessingFailed**        **0x8**
1465     The post-processing agent failed while trying to log accounting
1466     attributes for the job; therefore the job has been placed into
1467     the **completed** state with the **jobR~~r~~etained jm~~J~~obStateReasons1**
1468     object~~attribute~~ value for a system-defined period of time, so
1469     the administrator can examine it, resubmit it, etc. ~~The post-~~
1470     ~~processing agent is a plug and play mechanism which the system~~
1471     ~~and the customer uses to add functionality that is executed~~
1472     ~~after a job has finished processing.~~
1473
1474     **submissionInterrupted**        **0x10**
1475     Indicates that the job was not completely submitted for the
1476     following reasons: (1) the server has crashed before the job was
1477     closed by the client,~~. The server SHALL put the job into the~~
1478     ~~**completed** state (and SHALL not print the job).~~ (2) the server or
1479     the document transfer method has crashed in some non-recoverable
1480     way before the document data was entirely transferred to the
1481     server,~~. The server SHALL put the job into the **completed** state~~
1482     ~~(and SHALL not print the job).~~ (3) the client crashed or failed
1483     to close the job before the time-out period. Whether the server
1484     or device puts the job into the **pendingHeld** or **aborted** state
1485     depends on implementation.~~The server SHALL close the job and put~~
1486     ~~the job into the **held** state with **job-state-reasons** of~~
1487     ~~**submission-interrupted** and **job-hold-set** and with the job's **job-**~~
1488     ~~**hold** attribute set to **TRUE**. The user may release the job for~~
1489     ~~scheduling by issuing a job submission or management protocol~~
1490     ~~operation.~~
1491
1492     **maxJobFaultCountExceeded**        **0x20**
1493     The job has ~~been~~ faulted ~~and returned by the server~~ several
1494     times and has exceeded the administratively defined fault count
1495     limit~~that the **job-fault-count** exceeded the device's (or~~

1496   ~~server's, if not defined for the device)~~ **cfg-max-job-fault-**
1497   **count**.  ~~The job is automatically put into the~~ **held** ~~state~~
1498   ~~regardless of the~~ **hold-jobs-interrupted-by-device-failure**
1499   ~~attribute. This~~ **job-state-reasons** ~~value is used in conjunction~~
1500   ~~with the~~ **job-interrupted-by-device-failure** ~~value.~~
1501
1502   **devicesNeedAttentionTimeOut          0x40**
1503      One or more document transforms that the job is using needs
1504      human intervention in order for the job to make progress, but
1505      the human intervention did not occur within the site-settable
1506      time-out value ~~and the server/device has transitioned the job to~~
1507      ~~the~~ **held** ~~state~~.
1508
1509   **needsKeyOperatorTimeOut          0x80**
1510      One or more devices or document transforms that the job is using
1511      need a specially trained operator (who may need a key to unlock
1512      the device and gain access) in order for the job to make
1513      progress, but the key operator intervention did not occur within
1514      the site-settable time-out value ~~and the server/device has~~
1515      ~~transitioned the job to the~~ **held** ~~state~~.
1516
1517   **jobStartWaitTimeOut          0x100**
1518      The server/device has stopped the job at the beginning of
1519      processing to await human action, such as installing a special
1520      cartridge or special non-standard media, but the job was not
1521      resumed within the site-settable time-out value and the
1522      server/device has transitioned the job to the **pending~~H~~held**
1523      state.  Normally, the job is resumed by means outside the job
1524      submission protocol, such as some local function on the device.
1525
1526   **jobEndWaitTimeOut          0x200**
1527      The server/device has stopped the job at the end of
1528      **processing~~/printing~~** to await human action, such as removing a
1529      special cartridge or restoring standard media, but the job was
1530      not resumed within the site-settable time-out value and the
1531      server/device has transitioned the job to the **completed** state.
1532      Normally, the job is resumed by means outside the job submission
1533      protocol, such as some local function on the device, whereupon
1534      the job SHALL transition immediately to the **completed~~canceled~~**
1535      state.
1536
1537   **jobPasswordWaitTimeOut          0x400**
1538      The server/device has stopped the job at the beginning of
1539      processing to await input of the job's password, but the human
1540      intervention did not occur within the site-settable time-out
1541      value ~~and the server/device has transitioned the job to the~~ **held**
1542      ~~state.  Normally, the password is input and the job is resumed~~
1543      ~~by means outside the job submission protocol, such as some local~~
1544      ~~function on the device~~.

```
1545
1546        deviceTimedOut                          0x800
1547            A device that the job was using has not responded in a period
1548            specified by the device's site-settable attribute.
1549
1550        connectingToDeviceTimeOut               0x1000
1551            The server is attempting to connect to one or more devices which
1552            may be dial-up, polled, or queued, and so may be busy with
1553            traffic from other systems, but server was unable to connect to
1554            the device within the site-settable time-out value and the
1555            server has transitioned the job to the held state.
1556
1557        transferring                            0x2000
1558            The job is being transferred to a down stream server or device.
1559
1560        queuedInDevice                          0x4000
1561            The job has been queued in a down stream server or device.
1562
1563        jobCleanup                              0x8000
1564            The server/device is performing cleanup activity as part of
1565            ending normal processing.
1566
1567        processingToStopPoint                   0x10000
1568            The requester has issued an operation to interrupt the job and
1569            the server/device is processing up until the specified stop
1570            point occurs.
1571
1572        jobPasswordWait                         0x20000
1573            The server/device has selected the job to be next to process,
1574            but instead of assigning resources and started the job
1575            processing, the server/device has transitioned the job to the
1576            pendingHheld state to await entry of a password (and dispatched
1577            another job, if there is one).  The user resumes the job either
1578            locally or by issuing a remote operation and supplying a job-
1579            password=secret-code input parameter that must match the job's
1580            job-password attribute.
1581
1582        validating                              0x40000
1583            The server/device is validating the job after accepting the job.
1584            The job state may be held, pending, or processing.
1585
1586        queueHeld                               0x80000
1587            The operator has held the entire job set or queue by means
1588            outside the scope of the Job model.
1589
1590        jobProofWait                            0x100000
1591            The job has produced a single proof copy and is in the
1592            pendingHheld state waiting for the requester to issue an
1593            operation to release the job to print normally, obeying the any
```

1594        ~~job-copies~~ and ~~copy-count~~ job and document copy attributes that
1595        were originally submitted.
1596
1597    **heldForDiagnostics**                    **0x200000**
1598        The system is running intrusive diagnostics, so th~~e~~ate all jobs
1599        are being held.
1600
1601    **serviceOffLine**                        **0x400000**
1602        The service/document transform is off-line and accepting no
1603        jobs.  All **pending** jobs are put into the **pending~~H~~held** state.
1604        This could be true if its input is impaired or broken.
1605
1606    **noSpaceOnServer**                       **0x800000**
1607        T~~he job is held because t~~here is no room on the server to store
1608        all of the job.  For example, there is no room for the document
1609        data ~~or a scan to file job~~.
1610
1611    **pinRequired**                           **0x1000000**
1612        The System Administrator settable device policy is (1) to
1613        require PINs, and (2) to hold jobs that do not have a pin
1614        supplied as an input parameter when the job was created. The
1615        requester SHALL either (1) enter a pin locally at the device or
1616        issue a remote operation supplying the PIN in order for the job
1617        to be able to proceed.
1618
1619    **exceededAccountLimit**                  **0x2000000**
1620        The account for which this job is drawn has exceeded its limit.
1621        This condition SHOULD be detected before the job is scheduled so
1622        that the user does not wait until his/her job is scheduled only
1623        to find that the account is overdrawn.  This condition MAY also
1624        occur while the job is processing either as processing begins or
1625        part way through processing.
1626
1627        An overdraft mechanism SHOULD be included to be user-friendly,
1628        so as to minimize the chances that the job cannot finish or that
1629        media is wasted.  For example, the server/device SHOULD finish
1630        the current copy for a job with collated document copies, rather
1631        than stopping in the middle of the current document copy.
1632
1633    **heldForRetry**                          **0x4000000**
1634        The job encountered some errors that the server/device could not
1635        recover from with its normal retry procedures, but the error is
1636        worth trying the job later, such as phone number busy or remote
1637        file system in-accessible.  For such a situation, the
1638        server/device SHALL ~~add the **held-for-retry** value to the job's~~
1639        ~~**jobStateReasons2 attribute** and~~ transition the job from the
1640        **processing** to the **pending~~H~~held**, rather than to the
1641        **aborted~~completed~~** state.
1642

1643
1644    The following values are from the X/Open PSIS draft standard:
1645
1646        **canceledByShutdown**                    **0x8000000**
1647            The job was canceled because the server or device was shutdown
1648            before completing the job.  Whether the job is placed in the
1649            **pendingHeld** or **aborted** state, depends on implementation.The job
1650            SHALL be placed in the **pending** state [if the job was not
1651            started, else the job SHALL be placed in the **terminating** state].
1652
1653        **deviceUnavailable**                     **0x10000000**
1654            This job was aborted by the system because the device is
1655            currently unable to accept jobs.  Whether the job is placed in
1656            the **pendingHeld** or **aborted** state, depends on implementation.This
1657            reason [SHALL be] used in conjunction with the reason
1658            **abortedyystem**. The job SHALL be placed in the **pending** state.
1659
1660        **wrongDevice**                           **0x20000000**
1661            This job was aborted by the system because the device is unable
1662            to handle this particular job; the spooler SHOULD try another
1663            device or the user should submit the job to another device.
1664            Whether the job is placed in the **pendingHeld** or **aborted** state,
1665            depends on implementation.This reason [SHALL be] used in
1666            conjunction with the reason **abortedBySystem**.  The job SHALL be
1667            **pending** if the queue contains other physical devices that the
1668            job could print on, and the spooler is capable of not sending
1669            the job back to a physical device that has rejected the job for
1670            this job state reasons value. Otherwise, [the job] SHALL be
1671            placed in the **completed** state with the **jobRetained** value set in
1672            the **jobStateReasons1** attribute.
1673
1674        **badJob**                                **0x40000000**
1675            This job was aborted by the system because this job has a major
1676            problem, such as an ill-formed PDL; the spooler SHOULD not even
1677            try another device. This reason SHALL be used in conjunction
1678            with the reason **aborted** by system. The job SHALL be placed in
1679            the **terminating** state.
1680
1681    These bit definitions are the equivalent of a type 2 enum except that
1682    combinations of them may be used together.  See section 7.1.2 on page
1683    21."
1684
1685        SYNTAX        **INTEGER(0..2147483647)**   **--** 31 bits, all but sign bit
1686
1687
1688
1689
1690
1691

```
1692   JmJobStateReasons3TC ::= TEXTUAL-CONVENTION
1693        STATUS      current
1694        DESCRIPTION
1695            "This textual-convention is used with the jobStateReasons3
1696            attribute to provides additional information regarding the
1697            jmJobState/jobState object/attribute.  See the description under
1698            JmJobStateReasons1TC on page 60.
1699
1700            The following standard values are defined (in hexadecimal) as
1701            powers of two, since multiple values may be used at the same
1702            time:
1703
1704        jobInterruptedByDeviceFailure          0x1
1705            A device or the print system software that the job was using has
1706            failed while the job was processing.  The server or device is
1707            keeping the job in the pendingHheld state until an operator can
1708            determine what to do with the job.
1709
1710        These bit definitions are the equivalent of a type 2 enum except
1711        that combinations of them may be used together.  See section 7.1.2
1712        on page 21.  The remaining bits are reserved for future
1713        standardization and/or registration."
1714
1715        SYNTAX      INTEGER(0..2147483647)   -- 31 bits, all but sign bit
1716
1717
1718
1719
1720
1721   JmJobStateReasons4TC ::= TEXTUAL-CONVENTION
1722        STATUS      current
1723        DESCRIPTION
1724            "This textual-convention is used in the jobStateReasons4
1725            attribute to provides additional information regarding the
1726            jmJobState/jobState object/attribute.  See the description under
1727            JmJobStateReasons1TC on page 60.
1728
1729            The following standard values are defined (in hexadecimal) as
1730            powers of two, since multiple values may be used at the same
1731            time:
1732
1733        none yet defined.
1734
1735            These bit definitions are the equivalent of a type 2 enum except
1736            that combinations of them may be used together.  See section
1737            7.1.2 on page 21.     These bits are reserved for future
1738        standardization and/or registration."
1739
1740        SYNTAX      INTEGER(0..2147483647)   -- 31 bits, all but sign bit
```

1741
1742
1743
1744
1745

-- The following tables 1-4 show the **JmJobStateReasonsnTC** values
-- (*n*=1..4) and the job states for which they are applicable:

1746
1747  -- Table 1 - **JmJobStateReasons1TC**: Legal Job States for each Job State
1748  --                              Reason
1749

| -- Descriptive Name | Allowed job states |
|---|---|
| -- documents-needed(1) | held |
| -- job-hold-set(2) | held |
| -- job-process-after-specified(3) | held |
| -- required-resources-not-ready(4) | held |
| -- successful-completion(5) | completed |
| -- completed-with-warnings(6) | completed |
| -- completed-with-errors(7) | completed |
| -- canceled-by-user(8) | canceled |
| -- canceled-by-operator(9) | canceled |
| -- aborted-by-system(10) | canceled |
| -- logfile-pending(11) | canceled |
| -- logfile-transferring(12) | canceled |
| -- jobPreProcessing(45) | held |
| -- jobPaused(46) | held |
| -- jobInterrupted(47) | held |
| -- jobRetained(48) | canceled, completed |
| -- jobHoldUntilSpecified(49) | held |

1750

1751  -- Table 2 - **JmJobStateReasons2TC**: Legal Job States for each Job State
1752  --                              Reason
1753

| -- Descriptive Name | Allowed job states |
|---|---|
| -- cascaded(13) | canceled |
| -- deleted-by-administrator(14) | canceled |
| -- discard-time-arrived(15) | canceled |
| -- postprint-failed(16) | canceled, completed |
| -- submission-interrupted(17) | canceled |
| -- max-job-fault-count-exceeded(18) | canceled |
| -- devices-need-attention-time-out(19) | held, canceled |
| -- needs-key-operator-time-out(20) | held, canceled |
| -- job-start-wait-time-out(21) | canceled |

| — | Descriptive Name | Allowed job states |
|---|---|---|
| — | job-end-wait-time-out(22) | canceled |
| — | job-password-wait-time-out(23) | held, pending |
| — | device-timed-out(24) | held, canceled |
| — | connecting-to-device-time-out(25) | held, canceled |
| — | transferring(26) | processing |
| — | queued-in-device(27) | processing |
| — | job-cleanup(28) | processing |
| — | processing-to-stop-point(29) | processing |
| — | job-password-wait(30) | held, processing |
| — | validating(31) | held, pending, processing |
| — | queue-held(32) | held |
| — | job-proof-wait(33) | held |
| — | held-for-diagnostics(34) | held |
| — | service-off-line(35) | held |
| — | no-space-on-server(36) | held |
| — | pin-required(37) | held, canceled |
| — | exceeded-account-limit(38) | held, canceled |
| — | held-for-retry(39) | held |
| — | canceledByShutdown(40) | canceled |
| — | deviceUnavailable(41) | pending |
| — | wrongDevice(42) | canceled |
| — | badJob(43) | canceled |

1754

1755    — Table 3   **JmJobStateReasons3TC:** Legal Job States for each Job State
1756    ——————————————————————————— Reason
1757

| — | jobInterruptedByDeviceFailure(44) | held |

1758
1759

```
1760
1761   jobmonMIBObjects   OBJECT IDENTIFIER   ::= { jobmonMIB 1 }
1762
1763   -- The General Group (Mandatory)
1764
1765   -- The jmGeneralGroup consists entirely of the jmGeneralTable.
1766
1767   -- Implementation of every object in this group is MANDATORY.
1768   -- See Section 4 entitled 'Conformance Considerations' on page 18.
1769
1770   jmGeneral  OBJECT IDENTIFIER ::= { jobmonMIBObjects 15 }
1771
1772   jmGeneralTable  OBJECT-TYPE
1773       SYNTAX      SEQUENCE OF JmGeneralEntry
1774       MAX-ACCESS  not-accessible
1775       STATUS      current
1776       DESCRIPTION
1777           "The jmGeneralTable consists of information of a general nature
1778           that are per-job-set, but are not per-job.  See Terminology and
1779           Job Model on page 11 for the definition of a job set."
1780       ::= { jmGeneral 1 }
1781
1782   jmGeneralEntry  OBJECT-TYPE
1783       SYNTAX      JmGeneralEntry
1784       MAX-ACCESS  not-accessible
1785       STATUS      current
1786       DESCRIPTION
1787           "Information about a job set (queue).
1788
1789           An entry SHALL exist in this table for each job set."
1790       INDEX   { jmJobSetIndex }
1791       ::= { jmGeneralTable 1 }
1792
1793   JmGeneralEntry ::= SEQUENCE {
1794       jmGeneralNumberOfActiveJobs         Integer32(0..2147483647),
1795       jmGeneralOldestActiveJobIndex       Integer32(0..2147483647),
1796       jmGeneralNewestActiveJobIndex       Integer32(0..2147483647),
1797       jmGeneralJobPersistence             Integer32(0..2147483647),
1798       jmGeneralAttributePersistence       Integer32(0..2147483647),
1799       jmGeneralJobSetName                 OCTET STRING(SIZE(0..63))
1794   }
1795
1796   jmGeneralNumberOfActiveJobs OBJECT-TYPE
1797       SYNTAX      Integer32(0..2147483647)
1798       MAX-ACCESS  read-only
1799       STATUS      current
1800       DESCRIPTION
1801           "The current number of 'active' jobs in the jmJobIDTable,
1802           jmJobStateTable, and jmAttributeTable, i.e., the total number of
```

```
1803            jobs that are in the pending, processing, or processingStopped
1804            neither the completed nor the canceled states.  See JmJobStateTC
1805            on page 31 for the exact specification of the semantics of the
1806            job states.
1807
1808            If there are no active jobs, the value of this object SHALL be
1809            0."
1810        ::= { jmGeneralEntry 1 }
1811
1812    jmGeneralOldestActiveJobIndex  OBJECT-TYPE
1813        SYNTAX        Integer32 (0..2147483647)
1814        MAX-ACCESS   read-only
1815        STATUS        current
1816        DESCRIPTION
1817            "The jmJobIndex of the oldest active job that is still in one of
1818            the 'active' states (pending, processing, or processingStopped).
1819            In other words, the index of the 'active' job that has been in
1820            the , i.e., the job in the job tablesjmJobStateTable and
1821            jmAttributeTable that has been there the longest and .has
1822            neither completed nor been canceled.
1823
1824            When a job transitions from one of the 'active' states (pending,
1825            processing, processingStopped) to one of the 'in-active' states
1826            (pendingHeld, completeds, or is canceled, or aborted), with a
1827            jmJobIndex value that matches this object, the agent SHALL
1828            advance (or wrap - see jmGeneralNewestActiveJobIndex) the value
1829            to the next oldest 'active' job, if any.
1830
1831            On the other hand, when a job transitions from one of the 'in-
1832            active' states to one of the 'active' state, the agent SHALL
1833            reduce (or wrap) the value of this object, if the job's
1834            jmJobIndex is smaller than the current value.
1835
1836            If there are no active jobs, the agent SHALL set the value of
1837            this object to 0."
1838        ::= { jmGeneralEntry 2 }
1839
1840    jmGeneralNewestActiveJobIndex  OBJECT-TYPE
1841        SYNTAX        Integer32 (0..2147483647)
1842        MAX-ACCESS   read-only
1843        STATUS        current
1844        DESCRIPTION
1845            "The jmJobIndex of the newest active job that is in one of the
1846            'active' states (pending, processing, or processingStopped).  In
1847            other words, the index of the 'active' job that has been most
1848            recently added to the job tables., i.e., the job in the
1849            jmJobStateTable and jmAttributeTable that has been added most
1850            recently and has neither completed nor been canceled.
1851
```

1852     When a new job is accepted by the server or device that the
1853     agent is instrumenting, the agent SHALL assign the next
1854     available value to the job's **jmJobIndex** that is used for storing
1855     job information in the **jmJobIDTable**, the **jmJobTable**, and the
1856     **jmAttributeTable**.  ~~increment~~ this object **by 1** ~~and store the job~~
1857     ~~attributes in the row specified by the incremented value.~~  If
1858     the value would exceed the implementation-defined maximum value
1859     for **jmJobIndex**, the agent SHALL set the value back to **1**, i.e.,
1860     wrap around to the beginning of the job tables.
1861
1862     It is recommended that the largest value for **jmJobIndex** be much
1863     larger than the maximum number of jobs that the implementation
1864     can contain at a single time, so as to minimize the pre-mature
1865     re-use of **jmJobIndex** value for a newer job while clients retain
1866     the same 'stale' value for an older job.
1867
1868     Each time ~~When~~ a new job is accepted by the server or device
1869     that the agent is instrumenting AND that job is to be 'active'
1870     (**pending**, **processing**, or **processingStopped**, but not
1871     **pendingHeld**), the agent SHALL copy the value of the job's
1872     **jmJobIndex** to the **jmGeneralNewestActiveJobIndex** object.  If the
1873     new job is 'in-active' (**pendingHeld** state), the agent SHALL not
1874     change the value of **jmGeneralNewestActiveJobIndex** object.
1875
1876     When all jobs become 'inactive', i.e., enter the **pendingHeld,**
1877     **completed**, ~~or~~ **canceled, or aborted** states, the agent SHALL
1878     set ~~leave~~ the value of this object to **0** ~~unchanged~~.  Whenever a job
1879     changes from 'in-active' to 'active' (from **pendingHeld** to
1880     **pending** or **processing**), the agent SHALL update the value of
1881     either the **jmGeneralOldestActiveJobIndex** or the
1882     **jmGeneralNewestActiveJobIndex** objects, or both, if the job's
1883     **jmJobIndex** value is outside the range between
1884     **jmGeneralOldestActiveJobIndex** and **jmGeneralNewestActiveJobIndex.**
1885
1886     When the server or device is power-cycled, the agent SHALL
1887     remember the next **jmJobIndex** value to be assigned ~~the value of~~
1888     ~~this object shall be persistent~~, so that new jobs are not
1889     assigned the same **jmJobIndex** as recent jobs before the power
1890     cycle. ~~Therefore, the agent shall return the value 0 only on~~
1891     ~~the first power up of the server or device.~~
1892
1893     NOTE - Applications that wish to efficiently access all of the
1894     active jobs MAY use **jmGeneralOldestActiveJobIndex** value to start
1895     with the oldest active job and continue until they reach the
1896     index value equal to **jmGeneralNewestActiveJobIndex,** skipping
1897     over any **pendingHeld**, **completed**, ~~or~~ **canceled, or aborted** jobs
1898     that might intervene**.**
1899

```
1900            If an application detects that the jmGeneralNewestActiveJobIndex
1901            is smaller than jmGeneralOldestActiveJobIndex, the job index has
1902            wrapped.  In this case, when the application exceeds the maximum
1903            job index (detected by a no such object status returned from a
1904            GetNext operation for the next conceptual row), the application
1905            SHALL start over at 1 and continue the GetNext operations to
1906            find the rest of the active jobs."
1907        ::= { jmGeneralEntry 3 }
1908
1909    jmGeneralJobPersistence OBJECT-TYPE
1910        SYNTAX        Integer32(0..2147483647)
1911        MAX-ACCESS    read-only
1912        STATUS        current
1913        DESCRIPTION
1914            "The minimum time in seconds for this instance of the Job Set
1915            that an entry will remain in the jmJobIDTable and
1916            jmJobStateTable after processing has completed, i.e., the
1917            minimum time in seconds starting when the job enters the
1918            completed, canceled, or aborted state.  Depending on
1919            implementation, the value of this object MAY be either: (1) set
1920            by the system administrator by means outside this specification
1921            or (2) fixed by the implementation."
1922        ::= { jmGeneralEntry 4 }
1923
1924    jmGeneralAttributePersistence OBJECT-TYPE
1925        SYNTAX        Integer32(0..2147483647)
1926        MAX-ACCESS    read-only
1927        STATUS        current
1928        DESCRIPTION
1929            "The minimum time in seconds for this instance of the Job Set
1930            that an entry will remain in the jmAttributeTable after
1931            processing has completed , i.e., the time in seconds starting
1932            when the job enters the completed, canceled, or aborted state.
1933            The value of this object MAY be either (1) set by the system
1934            administrator by means outside this specification or MAY be (2)
1935            fixed by the implementation, depending on implementation.
1936
1937            This value SHALL be equal to or less than the value of
1938            jmGeneralJobPersistence."
1939        ::= { jmGeneralEntry 5 }
1940
1941    jmGeneralJobSetName OBJECT-TYPE
1942        SYNTAX        OCTET STRING(SIZE(0..63))
1943        MAX-ACCESS    read-only
1944        STATUS        current
1945        DESCRIPTION
1946            "The human readable administratively assigned name of this job
1947            set (by means outside of this MIB).  Typically, this name will
1948            be the name of the job queue.  If a server or device has only a
```

```
1949            single job set, this object can be the administratively assigned
1950            name of the server or device itself.  This name does not need to
1951            be unique, though each job set in a single Job Monitoring MIB
1952            SHOULD have distinct names.
1953
1954            NOTE - The purpose of this object is to help the user of the job
1955            monitoring application distinguish between several job sets in
1956            implementations that support more than one job set."
1957        ::= { jmGeneralEntry 6 }
1958
1959
1960
1961
1962
1963   -- The Job ID Group (Mandatory)
1964
1965   -- The jmJobIDGroup consists entirely of the jmJobIDTable.
1966   --
1967   -- The two key indexes that are used in other tables to index jobs:
1968   -- jmJobSetIndex and jmJobIndex are materialized in this group.
1969   --
1970   -- Implementation of every object in this group is MANDATORY.
1971   -- See Section 4 entitled 'Conformance Considerations' on page 18.
1972
1973   jmJobID  OBJECT IDENTIFIER ::= { jobmonMIBObjects 26 }
1974
1975   jmJobIDTable  OBJECT-TYPE
1976       SYNTAX       SEQUENCE OF JmJobIDEntry
1977       MAX-ACCESS   not-accessible
1978       STATUS       current
1979       DESCRIPTION
1980           "The jmJobIDTable provides a correspondence map (1) between the
1981           job submission ID that a client uses to refer to a job and (2)
1982           the jmJobSetIndex and jmJobIndex that the Job Monitoring MIB
1983           agent assigned to the job and that are used to access the job in
1984           all of the other tables in the MIB.  If a monitoring application
1985           already knows the jmJobIndex of the job it is querying, that
1986           application NEED NOT use the jmJobIDTable."
1987        ::= { jmJobID 1 }
1988
1989   jmJobIDEntry  OBJECT-TYPE
1990       SYNTAX       JmJobIDEntry
1991       MAX-ACCESS   not-accessible
1992       STATUS       current
1993       DESCRIPTION
1994           "The map from (1) the jmJobSubmissionID to (2) the jmJobSetIndex
1995           and jmJobIndex.
1996
```

```
1997          An entry SHALL exist in this table for each job, no matter what
1998          the state of the job and no matter what job set the job is in.
1999          Each job SHALL appear in one and only one job set.
2000
2001          NOTE - an IMPLICIT statement is NOT provided in the following
2002          INDEX clause, since it was not an SMIv1 feature.  Therefore, the
2003          extra ASN.1 tag SHALL be included in the varbind in the SNMP
2004          request and the response."
2005      INDEX  { jmJobSubmissionIDIndex }
2006      ::= { jmJobIDTable 1 }
2007
2008  JmJobIDEntry ::= SEQUENCE {
          jmJobSubmissionIDIndex              OCTET STRING(SIZE(10..32)),
          jmJobSetIndex                       Integer32(1..32767),
          jmJobIndex                          Integer32(1..2147483647)
2009  }
2010
2011  jmJobSubmissionIDIndex OBJECT-TYPE
2012      SYNTAX       OCTET STRING(SIZE(10..32))
2013      MAX-ACCESS   not-accessible
2014      STATUS       current
2015      DESCRIPTION
2016          "A quasi-unique 32-octet string ID which identifies the job
2017          uniquely within a particular client-server environment.  Either
2018          the client or the server assigns the job submission ID for each
2019          job.  The monitoring application whether in the client or
2020          running separately, uses the job submission ID to help the user
2021          identify which jmJobIndex was assigned by the agent.
2022
2023          There are multiple formats for the jmJobSubmissionIDIndex.  Each
2024          format SHALL be registered using the procedures of a type 2
2025          enum.  See section entitled: 'IANA Registration of enums' on
2026          page 21.
2027
2028          The value of jmJobSubmissionIDIndex SHOULD be one of the
2029          registered format types.  The first two octets of the string
2030          SHALL indicate which registered format is being used.  The ASCII
2031          characters '0-9', 'A-Z', and 'a-z' will be assigned in order
2032          giving 62 possible formats.  The agent SHALL assign a string of
2033          registered format (00) for any job without a Job Submission ID.
2034
2035          The format values registered so far are:
2036
2037          Format
2038          Number    Description
2039          ------    -----------
2040          00        Set by the agent when neither the client nor the
2041                    server assigned a job submission ID.
2042
```

```
2043        01         octets 3-10:  8-decimal-digit random number
2044                   octets 11-32: last 22 bytes of the jobName attribute
2045
2046        02         octets 3-10:  8-decimal-digit sequential number
2047                   octets 11-32: Client MAC address
2048
2049        03         octets 3-10:  8-decimal-digit sequential number
2050                   octets 11-32: last 22 bytes of the client URL
2051
2052        ..         to be registered according to procedures of a type 2
2053                   enum.  See section 7.3 on page 22.
2054
2055     NOTE - the job submission id is only intended to be unique
2056     between a limited set of clients for a limited duration of time,
2057     namely, for the life time of the job in the context of the
2058     server or device that is processing the job.  Some of the
2059     formats include something that is unique per client and a random
2060     number so that the same job submitted by the same client will
2061     have a different job submission id.  For other formats, where
2062     part of the id is guaranteed to be unique for each client, such
2063     as the MAC address or URL, a sequential number SHOULD suffice
2064     for each client (and may be easier for each client to manage).
2065     Therefore, the length of the job submission id has been selected
2066     to reduce the probability of collision to a very low number, but
2067     is not intended to be an absolute guarantee of uniqueness.
2068     None-the-less, collisions could occur, but without bad
2069     consequences, since this MIB is intended to be used only for
2070     monitoring jobs, not for controlling and managing them."
2071  ::= { jmJobIDEntry 1 }
2072
2073  jmJobSetIndex OBJECT-TYPE
2074     SYNTAX      Integer32(1..32767)
2075     MAX-ACCESS  read-only
2076     STATUS      current
2077     DESCRIPTION
2078        "The job set index of the job set in which the job was placed
2079        when that server or device accepted the job.  This 16-bit value
2080        in combination with the jmJobIndex value permits the management
2081        application to access the other tables to obtain the job-
2082        specific objects.  This value SHALL be the same for a job in the
2083        jmJobIDTable as the corresponding jmJobSetIndex value in the
2084        jmJobStateTable and jmAttributeTable for this job.
2085
2086        The value(s) of the jmJobSetIndex SHALL be persistent across
2087        power cycles, so that clients that have retained jmJobSetIndex
2088        values will access the same job sets upon subsequent power-up.
2089
2090        NOTE - aAn implementation that has only one job set, such as a
2091        printer with a single queue, SHALL hard code this object with
```

```
2092            the value 1.  See Terminology and Job Model on page 11 for the
2093            definition of a job set."
2094        ::= { jmJobIDEntry 2 }
2095
2096   jmJobIndex OBJECT-TYPE
2097        SYNTAX       Integer32(1..2147483647)
2098        MAX-ACCESS  read-only
2099        STATUS       current
2100        DESCRIPTION
2101            "The sequential, monatonically increasing identifier index for
2102            the job generated by the server or device when that server or
2103            device accepted the job.  This index value permits the
2104            management application to access the other tables to obtain the
2105            job-specific row entries.  This value SHALL be the index used in
2106            the jmJobStateTable and jmAttributeTable for this job.
2107
2108            See jmGeneralNewestActiveJobIndex on page 74 for a discussion
2109            about the largest value of jmJobIndex for an implementation.
2110
2111            NOTE    Agents instrumenting systems that contain jobs with a job
2112            identifier of 0 SHALL map the job identifier value 0 to a
2113            jmJobIndex value that is one higher than the highest job
2114            identifier value that any job can have on that system."
2115        ::= { jmJobIDEntry 3 }
2116
2117
2118
2119
2120   -- The Job State Group (Mandatory)
2121
2122   -- The jmJobStateGroup consists entirely of the jmJobStateTable.
2123   --
2124   -- Implementation of every object in this group is MANDATORY.
2125   -- See Section 4 entitled 'Conformance Considerations' on page 18.
2126
2127   jmJobStateG  OBJECT IDENTIFIER ::= { jobmonMIBObjects 37 }
2128
2129   jmJobStateTable  OBJECT-TYPE
2130        SYNTAX        SEQUENCE OF JmJobStateEntry
2131        MAX-ACCESS  not-accessible
2132        STATUS       current
2133        DESCRIPTION
2134            "The jmJobStateTable consists of basic job state and status
2135            information for each job in a job set that (1) monitoring
2136            applications need to be able to access in a single SNMP Get
2137            operation, (2) that have a single value per job, and (3) that
2138            SHALL always be implemented.
2139
```

```
2140      NOTE - Every accessible object in this table shall have the same
2141      value as one of the attributes in the jmAttributeTable.
2142      Implementations may either keep a separate copy or may share
2143      each value that is common between the jmJobStateTable and the
2144      jmAttributeTable.  The persistence of the two tables may be
2145      different depending on implementation and/or system
2146      administrator policy as specified by the jmGeneralJobPersistence
2147      and jmGeneralAttributePersistence objects defined on page .
2148      Thus an accounting application need only copy the entire
2149      jmAttributeTable or selected job rows and will obtain all of the
2150      information about those jobs and their states."
2151          ::= { jmJobStateG 1 }
2152
2153  jmJobStateEntry  OBJECT-TYPE
2154      SYNTAX       JmJobStateEntry
2155      MAX-ACCESS  not-accessible
2156      STATUS       current
2157      DESCRIPTION
2158          "Basic per-job state and status information.
2159
2160          An entry SHALL exist in this table for each job, no matter what
2161          the state of the job is.  Each job SHALL appear in one and only
2162          one job set."
2163      INDEX   { jmJobSetIndex, jmJobIndex }
2164          ::= { jmJobStateTable 1 }
2165
2166  JmJobStateEntry ::= SEQUENCE {
          jmJobState                         JmJobStateTC,        -- pg 31
          jmJobStateReasons1                 JmJobStateReasons1TC,  -- pg 60
          jmNumberOfInterveningJobs          Integer32(-2..2147483647),
          jmJobKOctetsRequested              Integer32(-2..2147483647),
          jmJobStateKOctetsProcessedCompleted Integer32(-2..2147483647),
          jmJobImpressionsRequested          Integer32(-2..2147483647),
          jmJobStateImpressionsCompleted     Integer32(-2..2147483647),
          jmJobStateAssociatedValue          Integer32(-2..2147483647)
2167  }
2168
2169
2170
2171  jmJobState OBJECT-TYPE
2172      SYNTAX       JmJobStateTC              -- See page 31
2173      MAX-ACCESS  read-only
2174      STATUS       current
2175      DESCRIPTION
2176          "The current state of the job (pending, processing, completed,
2177          etc.).  Even though the JmJobStateTC textual-convention defines
2178          nine values for job states, agents SHALL only implement those
2179          states which are appropriate for the particular implementation.
2180          In other words, all possible enums for this object SHALL be
```

```
2181            reported if implemented by the device and available to the
2182            agent.  However, management applications SHALL be prepared to
2183            receive all the standard job states.
2184
2185            The final value for this object SHALL be one of: completed,
2186            canceled, or aborted.  The minimum length of time that the agent
2187            SHALL keep a job in the completed, canceled, or aborted state
2188            before removing the job from the jmJobIDTable and jmJobTable is
2189            specified by the value of the jmGeneralJobPersistence object."
2190        ::= { jmJobStateEntry 1 }
2191
2192    jmJobStateReasons1 OBJECT-TYPE
2193        SYNTAX      JmJobStateReasons1TC                    -- See page 60
2194        MAX-ACCESS  read-only
2195        STATUS      current
2196        DESCRIPTION
2197            "Additional information about the job's current state, i.e.,
2198            information that augments the value of the job's
2199            jmJobState/jobState object/attribute.  NOTE - Companion textual
2200            conventions, JmJobStateReasonsnTC (n=1..4 - see page ) and
2201            corresponding attributes - see page  provides additional
2202            information about job states.
2203
2204            NOTE - The jobStateReasonsn (n=2..4) attributes (see page 41)
2205            provide further additional information about the job's current
2206            state.
2207
2208            Implementation of these values is OPTIONAL, i.e., an agent NEED
2209            NOT implement them, even if (1) the device supports the
2210            functionality represented by the reason and (2) is available to
2211            the agent.  These values MAY be used with any job state or
2212            states for which the reason makes sense.  Furthermore, when
2213            implemented, the agent SHALL return these values when the reason
2214            applies and SHALL NOT return them when the reason no longer
2215            applies whether the value of the job's jmJobState object changed
2216            or not.  When the job does not have any reasons for being in its
2217            current state, the agent SHALL set the value of the
2218            jmJobStateReasons1 object and jobStateReasonsn attributes to 0.
2219
2220            NOTE - While values cannot be added to the jmJobState object the
2221            job states cannot be added to without impacting deployed clients
2222            that take actions upon receiving jmJobState values, it is the
2223            intent that additional JmJobStateReasonsnTC enums can be defined
2224            and registered without impacting such deployed clients.  In
2225            other words, the jmJobStateReasons1 object and jobStateReasonsn
2226            attributes are intended to be extensible.The jobStateReasons1
2227            attribute identifies the reason or reasons that the job is in
2228            the held, pending, processing, needsAttention, canceled, or
2229            completed state.  The agent shall indicate the particular
```

```
2230        reason(s) by setting the value of the jobStateReasons1
2231        attribute.  "
2232     ::= { jmJobEntry 2 }
2233
2234 jmNumberOfInterveningJobs OBJECT-TYPE
2235     SYNTAX       Integer32(-2..2147483647)
2236     MAX-ACCESS   read-only
2237     STATUS       current
2238     DESCRIPTION
2239        "The number of jobs that are expected to be processed before
2240        this job is processed according to the implementation's queuing
2241        algorithm if no other jobs were to be submitted.  In other
2242        words, this value is the job's queue position.  The agent SHALL
2243        return a value of 0 for this attribute when this job starts
2244        processing (since there are no jobs in front of the job)."
2245     ::= { jmJobEntry 3 }
2246
2247 jmJobKOctetsRequested OBJECT-TYPE
2248     SYNTAX       Integer32(-2..2147483647)
2249     MAX-ACCESS   read-only
2250     STATUS       current
2251     DESCRIPTION
2252        "The total size innumber of K (1024) octets of the document(s)
2253        being requested to be processed in the job, including document
2254        and job copies.  The agent SHALL round the actual number of
2255        octets up to the next highest K.  Thus 0 octets SHALL be
2256        represented as 0, 1-1024 octets SHALL be represented as 1, 1025-
2257        2048 SHALL be represented as 2, etc.
2258
2259        The server/device MAY update the value of this attribute after
2260        each document has been transferred to the server/device or the
2261        server/device MAY provide this value after all documents have
2262        been transferred to the server/device, depending on
2263        implementation.  In other words, while the job is in the
2264        pendingHheld state with the jmJobStateReasons1 objectattribute
2265        containing a jobIncomingdocumentsNeeded or preProcessing value,
2266        the value of the jmJobKOctetsRequested objectattribute depends
2267        on implementation and MAY not correctly reflect the size of the
2268        job.
2269
2270        In computing this value, the server/device SHALL not include the
2271        multiplicative factors contributed by (1) the number of document
2272        copies, and (2) the number of job copies, independent of whether
2273        the device can process multiple copies of the job or document
2274        without making multiple passes over the job or document data and
2275        independent of  whether the output is collated or not.  Thus the
2276        server/device computation is independent of the implementation.
2277        and shall be:
2278
```

2279          (1) Document contribution:  Multiply the size of each document
2280          in octets by the number of document copies of that document.
2281
2282          (2) Add each document contribution together.
2283
2284          (3) Job copy contribution:  Multiply the job size by the number
2285          of job copies.
2286
2287          (4) Round up the result to the next higher K (1024 multiple)."
2288      ::= { jmJobEntry 4 }
2289
2290  **jmJobStateKOctetsProcessedCompleted** OBJECT-TYPE
2291      SYNTAX      **Integer32(-2..2147483647)**
2292      MAX-ACCESS  read-only
2293      STATUS      current
2294      DESCRIPTION
2295          "The current number of octets completed processeding by the
2296          server or device measured in units of K (1024) octets.  The
2297          agent SHALL round the actual number of octets processedcompleted
2298          up to the next higher K.  Thus 0 octets SHALL be is represented
2299          as **0**, 1-10243 octets, SHALL be is represented as **1**, 10254-20487
2300          octets SHALL be is **2**, etc. — For printing devices, this value is
2301          the number interpreted by the page description language
2302          interpreter rather than what has been marked on media.
2303
2304          For implementations where multiple copies are produced by the
2305          interpreter makes only a single pass over the document, the
2306          final value SHALL be equal to the value of the
2307          **jmJobKOctetsRequested** object.  For implementations where
2308          multiple copies are produced by the interpreter making multiple
2309          passes over the document, the final value SHALL be a multiple of
2310          the value of the **jmJobKOctetsRequested** object.The value of this
2311          object shall always be the same as that of the
2312          **jobKOctetsCompleted** attribute, so that this information appears
2313          in both the **jmJobStateTable** and the **jmAttributeTable**
2314          simultaneously.  See the **jobKOctetsCompleted** attribute on page
2315          in the **jmAttributeTable** for the full specification of this
2316          object/attribute.
2317
2318          NOTE - See the **impressionsCompletedCurrentCopy** and
2319          **pagesCompletedCurrentCopy** attributes for attributes that areis
2320          reset on each document copy.
2321
2322          NOTE - The **jmJobKOctetsProcessedCompleted** object can be used in
2323          the numerator with the **jmJobKOctetsRequested** objectattribute in
2324          the denominator in order to produce a "thermometer" that
2325          indicates the progress of the job, provided that the
2326          multiplicative factor is taken into account for some
2327          implementations of multiple copies."

```
2328        ::= { jmJobStateEntry 52 }
2329
2330    jmJobImpressionsRequested OBJECT-TYPE
2331        SYNTAX      Integer32(-2..2147483647)
2332        MAX-ACCESS  read-only
2333        STATUS      current
2334        DESCRIPTION
2335            "The number of impressions requested by this job to produce."
2336        ::= { jmJobEntry 6 }
2337
2338    jmJobStateImpressionsCompleted OBJECT-TYPE
2339        SYNTAX      Integer32(-2..2147483647)
2340        MAX-ACCESS  read-only
2341        STATUS      current
2342        DESCRIPTION
2343            "The current number of impressions completed being marked and
2344            stacked by the device for this job so far.  For printing
2345            devices, the impressions completed includes interpreting,
2346            marking, and stacking the output.  For other types of job
2347            services, the number of impressions completed includes the
2348            number of impressions processed.
2349
2350            The value of this object shall always be the same as that of the
2351            impressionsCompleted attribute, so that this information appears
2352            in both the jmJobStateTable and the jmAttributeTable
2353            simultaneously.  See the impressionsCompleted attribute on page
2354            in the jmAttributeTable for the full specification of this
2355            object/attribute."
2356        ::= { jmJobStateEntry 73 }
2357
2358    jmJobStateAssociatedValue OBJECT TYPE
2359        SYNTAX      Integer32(-2..2147483647)
2360        MAX-ACCESS  read-only
2361        STATUS      current
2362        DESCRIPTION
2363            "The value of the most relevant attribute associated with the
2364            job's current state.
2365
2366
2367
2368
2369    -- The Attribute Group (Mandatory)
2370
2371    -- The jmAttributeGroup consists entirely of the jmAttributeTable.
2372    --
2373    -- Implementation of the twoevery objects in this group is MANDATORY.
2374    -- See Section 4 entitled 'Conformance Considerations' on page 18.
2375    --
```

```
2376   -- A fewSome attributes are MANDATORY for agent conformance, and the
2377   rest
2378   -- aare
2379   ---OPTIONALconditionally mandatory.  See the specification of the
2380   ---JmAttributeTypeTC on
2381   -- page 35 for which attributes are MANDATORY for
2382   ---agents to implement.
2383
2384   jmAttribute  OBJECT IDENTIFIER ::= { jobmonMIBObjects 48 }
2385
2386   jmAttributeTable  OBJECT-TYPE
2387       SYNTAX      SEQUENCE OF JmAttributeEntry
2388       MAX-ACCESS  not-accessible
2389       STATUS      current
2390       DESCRIPTION
2391           "The jmAttributeTable SHALL contain attributes of the job and
2392           document(s) for each job in a job set.  Instead of allocating
2393           distinct objects for each attribute, each attribute is
2394           represented as a separate row in the jmAttributeTable.  Some
2395           attributes represent information about the job and document(s),
2396           such as file names, document names, submission time, completion
2397           time, size, etc.  Other attributes represent requested and/or
2398           consumed resources for each job for use by monitoring and
2399           accounting applications."
2400       ::= { jmAttribute 1 }
2401
2402   jmAttributeEntry  OBJECT-TYPE
2403       SYNTAX      JmAttributeEntry
2404       MAX-ACCESS  not-accessible
2405       STATUS      current
2406       DESCRIPTION
2407           "Attributes representing information about the job and
2408           document(s) or resources required and/or consumed.
2409
2410           Each entry in the jmAttributeTable is a per-job entrytable with
2411           an extra index for each type of attribute (jmAttributeTypeIndex)
2412           that a job can have and an additional index
2413           (jmAttributeInstanceIndex) for those attributes that can have
2414           multiple instances per job.  The jmAttributeTypeIndex object
2415           SHALL contain an enum type that indicates the type of attribute
2416           (see JmAttributeTypeTC on page 35).  The value of the attribute
2417           SHALL be represented in either the jmAttributeValueAsInteger or
2418           jmAttributeValueAsOctets objects, and/or both, as specified in
2419           the JmAttributeTypeTC textual-convention.
2420
2421           The agent SHALL create rows in the jmAttributeTable as the
2422           server or device is able to discover the attributes either from
2423           the job submission protocol itself or from the document PDL.  As
2424           the documents are interpreted, the interpreter MAY discover
```

2425          additional attributes and so the agent adds additional rows to
2426          this table.  As the attributes that represent resources are
2427          actually consumed, the usage counter contained in the
2428          **jmAttributeValueAsInteger** object is incremented according to the
2429          units indicated in the description of the **JmAttributeTypeTC**
2430          enum.
2431
2432          The agent SHALL maintain each row in the **jmJobTable** for at least
2433          the minimum time after a job completes as specified by the
2434          **jmGeneralAttributePersistence** (see page 76).
2435
2436          Zero or more entries SHALL exist in this table for each job in a
2437          job set.  Each job SHALL appear in one and only one job set."
2438     INDEX  { **jmJobSetIndex**, **jmJobIndex**, **jmAttributeTypeIndex**,
2439     **jmAttributeInstanceIndex** }
2440     ::= { jmAttributeTable 1 }
2441
2442  JmAttributeEntry ::= SEQUENCE {
          **jmAttributeTypeIndex**          **JmAttributeTypeTC**,          -- pg 35
          **jmAttributeInstanceIndex**       **Integer32(1..32767)**,
          **jmAttributeValueAsInteger**      **Integer32(-2..2147483647)**,
          **jmAttributeValueAsOctets**       **OCTET STRING(SIZE(0..63))**
2443  }
2444
2445  **jmAttributeTypeIndex** OBJECT-TYPE
2446     SYNTAX      **JmAttributeTypeTC**          -- See page 35
2447     MAX-ACCESS  not-accessible
2448     STATUS      current
2449     DESCRIPTION
2450          "The type of attribute that this row entry represents.
2451
2452          The type MAY identify information about the job or document(s)
2453          or MAY identify a resource required to process the job before
2454          the job start processing and/or consumed by the job as the job
2455          is processed.
2456
2457          Examples of job and document attributesinformation include:
2458          **jobCopiesRequested**, **documentCopiesRequested**, **jobCopiesCompleted**,
2459          **documentCopiesCompleted**, **fileName**, and **documentName**.
2460
2461          Examples of resources required and consumed resource attributes
2462          include:  **jobKOctetsRequested, jobKOctetsCompleted,**
2463          **pagesRequested**, **pagesCompleted**, **mediumRequested**, and
2464          **mediumConsumed,** respectively.
2465
2466          In the definitions of the enums in the **JmAttributeTypeTC** textual
2467          convention, each description indicates whether the value of the
2468          attribute shall be represented using the
2469          **jmAttributeValueAsInteger** or the **jmAttributeValueAsOctets**

```
2470        objects by the initial tag: 'INTEGER:' or 'OCTETS:',
2471        respectively.  A very few attributes use both objects
2472        (mediumConsumed) and so have both tags.
2473
2474        If the jmAttributeValueAsInteger object is not used (no
2475        'INTEGER:' tag), the agent shall return the value (-1)
2476        indicating other.  If the jmAttributeValueAsOctets object is not
2477        used (no 'OCTETS:' tag), the agent shall return a zero length
2478        octet string."
2479    ::= { jmAttributeEntry 1 }
2480
2481 jmAttributeInstanceIndex OBJECT-TYPE
2482    SYNTAX        Integer32(1..32767)
2483    MAX-ACCESS  not-accessible
2484    STATUS        current
2485    DESCRIPTION
2486        "A running 16-bit index of the attributes of the same type for
2487        each job.  For those attributes with only a single instance per
2488        job, this index value SHALL be 1.  For those attributes that are
2489        a single value per document, the index value SHALL be the
2490        document number, starting with 1 for the first document in the
2491        job.  Jobs with only a single document SHALL use the index value
2492        of 1.  For those attributes that can have multiple values per
2493        job or per document, such as documentFormatIndex or
2494        documentFormatType, the index SHALL be a running index for the
2495        job as a whole, starting at 1."
2496    ::= { jmAttributeEntry 2 }
2497
2498 jmAttributeValueAsInteger OBJECT-TYPE
2499    SYNTAX        Integer32(-2..2147483647)
2500    MAX-ACCESS  read-only
2501    STATUS        current
2502    DESCRIPTION
2503        "The integer value of the attribute.  The value of the attribute
2504        SHALL be represented as an integer if the enum description in
2505        the JmAttributeTypeTC definition (see page 35) has the tag:
2506        'INTEGER:'.
2507
2508        Depending on the enum definition, this object value MAY be an
2509        integer, a counter, an index, or an enum, depending on the
2510        jmAttributeTypeIndex value.  The units of this value are
2511        specified in the enum description.
2512
2513        For those attributes that are accumulating job consumption as
2514        the job is processed as specified in the JmAttributeTypeTC,
2515        SHALL contain the final value after the job completes
2516        processing, i.e., this value SHALL indicate the total usage of
2517        this resource made by the job.
2518
```

2519          A monitoring application is able to copy this value to a
2520          suitable longer term storage for later processing as part of an
2521          accounting system.
2522
2523          Since the agent MAY add attributes representing resources to
2524          this table while the job is waiting to be processed or being
2525          processed, which can be a long time before any of the resources
2526          are actually used, the agent SHALL set the value of the
2527          **jmAttributeValueAsInteger** object to **0** for resources that the job
2528          has not yet consumed.
2529
2530          Attributes for which the concept of an integer value is
2531          meaningless, such as **fileName**, **interpreter,** and
2532          **physicalDevice**~~Name~~**,** do *not* have the 'INTEGER:' tag in the
2533          **JmAttributeTypeTC** definition and so SHALL return a value of (**-1**)
2534          to indicate **other** for **jmAttributeValueAsInteger**.
2535
2536          For attributes which do have the 'INTEGER:' tag in the
2537          **JmAttributeTypeTC** definition, if the integer value is not (yet)
2538          known, the value SHALL be (**-2**) to represent unknown counting
2539          integers, **(2) to represent unknown enum values,** or the attribute
2540          row SHALL not be present in the table."
2541       ::= { jmAttributeEntry 3 }
2542
2543  **jmAttributeValueAsOctets** OBJECT-TYPE
2544       SYNTAX        **OCTET STRING(SIZE(0..63))**
2545       MAX-ACCESS  read-only
2546       STATUS        current
2547       DESCRIPTION
2548          "The octet string value of the attribute.  The value of the
2549          attribute SHALL be represented as an OCTET STRING if the enum
2550          description in the **JmAttributeTypeTC** definition (see page 35)
2551          has the tag: 'OCTETS:'.
2552
2553          Depending on the enum definition, this object value MAY be a
2554          coded character set string (text) or a binary octet string, such
2555          as **DateAndTime**.
2556
2557          Attributes for which the concept of an octet string value is
2558          meaningless, such as **pagesCompleted**, do *not* have the tag
2559          'OCTETS:' in the **JmAttributeTypeTC** definition and so the agent
2560          SHALL return ~~a value of~~ a zero length string for the value of
2561          the **jmAttributeValueAsOctets** object."
2562       ::= { jmAttributeEntry 4 }
2563

```
2564   -- Notifications and Trapping
2565   -- Reserved for the future
2566
2567   jobmonMIBNotifications   OBJECT IDENTIFIER  ::= { jobmonMIB 2}
2568
2569
2570
2571   -- Conformance Information
2572
2573   jmMIBConformance OBJECT IDENTIFIER ::= { jobmonMIB 32 }
2574
2575   -- compliance statements
2576   jmMIBCompliance MODULE-COMPLIANCE
2577       STATUS  current
2578       DESCRIPTION
2579           "The compliance statement for agents that implement the
2580           job monitoring MIB."
2581       MODULE -- this module
2582       MANDATORY-GROUPS {
2583           jmGeneralGroup, jmJobIDGroup, jmJobStateGroup, jmAttributeGroup
2584           }
2585
2586       OBJECT  jmJobState
2587       SYNTAX      INTEGER {
                           processing(5),
                           needsAttention(7),
                           canceled(8),
                           completed(9)
2588       }
2589       DESCRIPTION
2590           "It is conformant for an agent to implement just these four
2591           states in this object.  Any additional states are conditionally
2592           mandatory, i.e., an agent shall represent any additional states
2593           that the server or device implements.  However, a client shall
2594           accept all of the states from an agent."
2595
2596       -- OBJECT jmAttributeTypeIndex
2597       -- SYNTAX      INTEGER {
2598       --     jobOwner(2015)
2599       -- }
2600   -- DESCRIPTION
2601       --"It is conformant for an agent to implement just the one
2602       mandatory
2603       these 8
2604       -- attributes.  Any additional attributes are
2605       OPTIONAL,conditionally
2606       -- mandatory, i.e., an agent NEED NOTshall represent any
2607       additional
```

```
2608 |          -- attributesstates that the server or device implements.
2609              However, a
2610              -- client SHALL accept all of the attributes from an agent and
2611              -- either display them to its user or ignore them.
2612              --
2613              -- NOTE - SMI does not allow an enum to be declared as mandatory
2614              -- if that enum is not a member of a group, but
2615              -- jmAttributeTypeIndex cannot be a member of a group and still
2616 |            -- be not-accessible.  So this MIB spec comments the MANDATORY
2617              -- attributes as if SMI allowed such a declaration in order to
2618              -- declare the MANDATORY attributes."
2619
2620      -- There are no CONDITIONALLY MANDATORY or OPTIONAL groups.
2621
2622          ::= { jmMIBConformance 1 }
2623
2624      jmMIBGroups      OBJECT IDENTIFIER ::= { jmMIBConformance 2 }
2625
2626      jmGeneralGroup OBJECT-GROUP
2627          OBJECTS {
2628              jmGeneralNumberOfActiveJobs, jmGeneralOldestActiveJobIndex,
2629              jmGeneralNewestActiveJobIndex, jmGeneralJobPersistence,
2630              jmGeneralAttributePersistence, jmGeneralJobSetName}
2631          STATUS  current
2632          DESCRIPTION
2633              "The general group."
2634          ::= { jmMIBGroups 1 }
2635
2636      jmJobIDGroup OBJECT-GROUP
2637          OBJECTS {
2638              jmJobSetIndex, jmJobIndex }
2639          STATUS  current
2640          DESCRIPTION
2641              "The job ID group."
2642          ::= { jmMIBGroups 2 }
2643
2644 |    jmJobStateGroup OBJECT-GROUP
2645          OBJECTS {
2646 |            jmJobState, jmJobStateReasons1, jmNumberOfInterveningJobs,
2647 |            jmJobKOctetsRequested, jmJobStateKOctetsProcessedCompleted,
2648 |            jmJobImpressionsRequested, jmJobStateImpressionsCompleted,
2649 |            jmJobStateAssociatedValue }
2650          STATUS  current
2651          DESCRIPTION
2652 |            "The job state group."
2653          ::= { jmMIBGroups 3 }
2654
2655      jmAttributeGroup OBJECT-GROUP
2656          OBJECTS {
```

```
2657              jmAttributeValueAsInteger, jmAttributeValueAsOctets }
2658      STATUS  current
2659      DESCRIPTION
2660          "The attribute group."
2661      ::= { jmMIBGroups 4 }
2662
2663
2664  END
```

2665 ## 12.  Appendix A - Instrumenting the Job Life Cycle

2666 The job object has well-defined states and client operations that affect the transition between the
2667 job states.  Internal server and device actions also affect the transitions of the job between the job
2668 states.  These states and transitions are referred to as the job's *life cycle*.

2669 Not all implementations of job submission protocols have all of the states of the job model
2670 specified here.  The job model specified here is intended to be a superset of most implementations.
2671 It is the purpose of the agent to map the particular implementation's job life cycle onto the one
2672 specified here.  The agent MAY omit any states not implemented.  Only the **processing**,
2673 **needsAttention, canceled**, **aborted,** and **completed** states are required to be implemented by an
2674 agent.  However, a conforming management application SHALL be prepared to accept any of the
2675 states in the job life cycle specified here, so that the management application can interoperate with
2676 any conforming agent.

2677 The job states are intended to be the user visible.  The agent SHALL make these states visible in
2678 the MIB, but only for the subset of job states that the implementation has.  Implementations MAY
2679 need to have sub-states of these user-visible states.  Such implementation is *not* specified in this
2680 model, is not supported by this Job Monitoring MIB, and will vary from implementation to
2681 implementation.  In some implementations the **jmJobStateReasons1** object and the
2682 **jobStateReasons*n*** (*n*=2..4) attributes MAY represent some or all of the sub-states of the jobs.

2683 One of the purposes of the job life cyclemodel is to specify what is invariant from implementation
2684 to implementation as far as the MIB specification and the management application user is
2685 concerned.  Therefore, job states are all intended to last a user-visible length of time in most
2686 implementations.  However, some jobs may pass through some states in zero time in some
2687 situations and/or in some implementations.

2688 The job model does not specify how accounting and auditing is implemented, except to
2689 assumerequire that accounting and auditing logs are separate from the job life cycle and last
2690 longer than job entries in the MIBobjects.  Jobs in the **completed, aborted,** or **canceled** states are
2691 not logs, since jobs in these **completed** states are accessible via SNMPjob submission and/or job
2692 management protocol operations and SHALL beare removed from these Job Monitoring MIB
2693 tables after a site-settable or implementation-defined period of time.  An accounting application
2694 MAY copy Aaccounting information may be copied incrementally to anthe accounting logs as a
2695 job processes, or MAY be copied while the job is in the **canceled, aborted,** or **completed** states,
2696 depending on implementation.  The same is true for auditing logs.

2697 **The jmJobState object and the jobState attribute both specifiesy the standard job states.**
2698 **The normallegal job state transitions are shown in the state transition diagram presented in**
2699 **Table 1.  An implementation need not support all legal job state transitions.**

| New State |  |  |
|---|---|---|
| **"active" jobs** |  |  |

| Old state | unknown 2 | held 3 | pending 4 | processing 5 | printing 6 | needsAttention 7 | canceled 8 | completed 9 |
|---|---|---|---|---|---|---|---|---|
| unknown(2) | | yes | yes | yes | yes | | | |
| held(3) | | | yes | yes | yes | | yes | |
| pending(4) | | yes | | yes | yes | | yes | |
| processing(5) | | yes | | | yes | yes | yes | yes |
| printing(6) | | yes | | | | yes | yes | yes |
| needsAttention(7) | | yes | | yes | yes | | yes | |
| canceled(8) | yes | | | | | | | |
| completed(9) | yes | | | | | | | |

2700

2701   **13.   APPENDIX B - Support of the Job Submission ID in Job**
2702   **Submission Protocols**

2703   This appendix lists the job submission protocols that support the concept of a job
2704   submission ID and indicates the attribute in that protocol.

2705   **13.1   Hewlett-Packard's Printer Job Language (PJL)**

2706   Hewlett-Packard's Printer Job Language provides job-level printer control and printer
2707   status information to applications. The PJL JOB command is used at the beginning of a
2708   print job and can include options applying only to that job. A PJL JOB command option
2709   has been defined to facilitate passing the **JobSubmissionID** with the print job, as required
2710   by the Job Monitoring MIB. The option is of the form:
2711
2712         **SUBMISSIONID = "id string"**
2713

2714   Where the "id string" is a string and must be enclosed in double quotes.  The format is as
2715   described for the **jmJobSubmissionID** object.

2716   The entire PJL JOB command with the optional parameter would be of the form:
2717
2718         **@PJL JOB SUBMISSIONID = "id string"**
2719

2720   See "Printer Job Language Technical Reference Manual", part number 5021-0328, from
2721   Hewlett-Packard for complete information on the PJL JOB command and the Printer Job
2722   Language.

## 14. Bibliography

2723

[1] The Printer MIB - RFC 1579, proposed IETF standard.  Also an Internet-Draft on the standards track as a draft standard: **draft-ietf-printmib-mib-info-01.txt**

[2] ISO/IEC 10175 Document Printing Application (DPA).  See **ftp://ftp.pwg.org/pub/pwg/dpa/**

[3] Internet Printing Protocol (IPP), in progress on the IETF standards track.  See **draft-ietf-ipp-model-010.txt**.  See also **http://www.pwg.org/ipp/index.html**

[4] IEEE 1284.1, Transport-independent Printer System Interface (TIPSI).

[5] MIB-II, RFC 1213.

[6] Host Resources MIB, RFC 1514

[7] RFC 2119

## 15. Author's Addresses

Ron Bergman
Dataproducts Corp.
1757 Tapo Canyon Road
Simi Valley, CA 93063-3394

Phone: 805-578-4421
Fax:  805-578-4001
Email: rbergman@dpc.com


Tom Hastings
Xerox Corporation, ESAE-231
701 S. Aviation Blvd.
El Segundo, CA   90245

Phone: 310-333-6413
Fax:   310-333-5514
EMail: hastings@cp10.es.xerox.com


Scott A. Isaacson
Novell, Inc.
122 E 1700 S
Provo, UT   84606

2759
2760          Phone: 801-861-7366
2761          Fax:   801-861-4025
2762          EMail: scott_isaacson@novell.com
2763
2764
2765          Harry Lewis
2766          IBM Corporation
2767          6300 Diagonal Hwy
2768          Boulder, CO 80301
2769
2770          Phone: (303) 924-5337
2771          Fax:
2772          Email: harryl@us.ibm.com
2773
2774
2775          Send comments to the printmib WG using the Job Monitoring Project (JMP)
2776          Mailing List:
2777          JMP Mailing List:  jmp@pwg.org
2778
2779          To learn how to subscribe, send email to: JMP Mailing List Subscription
2780          Information:
2781          jmp-request@pwg.org
2782
2783          For further information, access the PWG web page under "JMP":
2784          http://www.pwg.org/
2785

2786    Other Participants:

2787          Chuck Adams - Tektronix
2788          Jeff Barnett - IBM
2789          Keith Carter, IBM Corporation
2790          Jeff Copeland - QMS
2791          Andy Davidson - Tektronix
2792          Roger deBry - IBM
2793          Mabry Dozier - QMS
2794          Lee Ferrel - Canon
2795          Steve Gebert - IBM
2796          Robert Herriot - Sun Microsystems Inc.
2797          Shige Kanemitsu - Kyocera
2798          David Kellerman - Northlake Software

| | |
|---|---|
| 2799 | Rick Landau - Digital |
| 2800 | Harry Lewis - IBM |
| 2801 | Pete Loya - HP |
| 2802 | Ray Lutz - Cognisys |
| 2803 | Jay Martin - Underscore |
| 2804 | Mike MacKay, Novell, Inc. |
| 2805 | Stan McConnell - Xerox |
| 2806 | Carl-Uno Manros, Xerox, Corp. |
| 2807 | Pat Nogay - IBM |
| 2808 | Bob Pentecost - HP |
| 2809 | Rob Rhoads - Intel |
| 2810 | David Roach - Unisys |
| 2811 | Hiroyuki Sato - Canon |
| 2812 | Bob Setterbo - Adobe |
| 2813 | Gail Songer, EFI |
| 2814 | Mike Timperman - Lexmark |
| 2815 | Randy Turner - Sharp |
| 2816 | William Wagner - Digital Products |
| 2817 | Jim Walker - Dazel |
| 2818 | Chris Wellens - Interworking Labs |
| 2819 | Rob Whittle - Novell |
| 2820 | Don Wright - Lexmark |
| 2821 | Lloyd Young - Lexmark |
| 2822 | Atsushi Yuki - Kyocera |
| 2823 | Peter Zehler, Xerox, Corp. |

2824  **16.  INDEX**

2825   This index includes the textual conventions, the objects, and the attributes.  Textual
2826   conventions all start with the prefix:  "**JM**" and end with the suffix:  "**TC**".  Objects all
2827   starts with the prefix:  "**jm**" followed by the group name.  Attributes are identified with
2828   enums, and so start with any lower case letter and have no special prefix.

2964