Job Monitoring MIB

From: Tom Hastings
Date: 03/26/97
Version: 0.71
File: ftp://ftp.pwg.org/pub/jmp/mibs/jmp-mib.doc .pdf .pdr

Status: Third draft MIB that corresponds to the sixth draft spec as agreed at the 02/07/97 JMP meeting and subsequent telecons. This is version 0.71. There are just a few changes from version 0.7, mostly editorial. See the change history.

The MIB has been greatly simplified so that now there are only 27 objects in the MIB: 21 mandatory and 6 conditionally mandatory.

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 telecon.

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.

The greatly simplified specifications of each object is derived from the ISO DPA attribute specifications in most cases. I've moved the full ISO DPA specifications to an Appendix. Revision marks show the agreements reached at the November meeting where we were able to finish the entire document. I've indicated ISSUES in the text that we have identified as issues but have not resolved. These issues are also listed at the end of the Table of Contents with the page number of the issue. I've also copied in map-summ.doc into this document and moved it to an appendix so we can more easily 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.

I've moved Ron's re-written introduction into the document.

INTERNET-DRAFT

Ron Bergman
Dataproducts Corp.
Tom Hastings
Xerox Corporation
Scott Isaacson
Novell, Inc.
Harry Lewis
IBM Corp.
March 1997

Job Monitoring MIB - V0.7

<draft-ietf-printmib-job-monitor-00.txt>

Expires Sept 26, 1997

Abstract

This Internet-Draft specifies a set of SNMP MIB objects for (1) monitoring the status and progress of print jobs (2) obtaining resource requirements before a job is processed, (3) monitoring resource consumption while a job is being processed and (4) collecting resource accounting data after the completion of a job. This MIB is intended to be implemented in printers or a server that supports one or more printers. Use of the object set is not limited to printing. However, support for services other than printing is outside the scope of this Job Monitoring MIB. Future extensions to this MIB

```
    may include, but are not limited to, fax machines and
    scanners.
```

TABLE OF CONTENTS

TABLE OF FIGURES

TABLE OF TEXTUAL-CONVENTIONS

Job Monitoring MIB

1. Introduction

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

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

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

User:

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

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

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

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

Operator:

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

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

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

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

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

Capacity Planner:

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

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

Accountant:

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

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

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

The MIB provides job resource accounting information after the printer has finished printing the job. This resource accounting information is intended to be used by:

- A management station that is co-located with the printer to provide an enhanced console capability.
- End user job monitoring programs that provide status on progress and completion of jobs during the complete life cycle of the job, including a defined period after the job completes.
- System accounting programs that copy the completed job statistics to an accounting system. It is recognized that depending on accounting programs to copy MIB data during the job-retention period is somewhat unreliable, since the accounting program may not be running (or may have crashed).

The MIB provides a set of objects that represent a compatible subset of job and document attributes of the ISO DPA standard, so that coherence is maintained between the two protocols and information presented to end users and system operators. However, the job monitoring MIB is intended to be used with printers that implement other job submitting and management protocols, such as IEEE 1284.1 (TIPSI), as well as with ones that do implement ISO DPA. So nothing in the job monitoring MIB shall require implementation of the ISO DPA protocol.

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

2. Terminology and Job Model

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

NOTE - Existing systems use conflicting terms, so these terms are drawn from the ISO 10175 Document Printing Application (DPA) standard. For example, PostScript systems use the term *session* for what we call a *job* in this specification and the term *job* to mean what we call a *document* in this paper. PJL systems use the term ..

A *job* is a unit of work whose results are expected together without interjection of unrelated results. A *client* is able to specify *job instructions* that apply to the job as a whole. Proscriptive instructions specify how, when, and where the job is to be printed. Descriptive instructions describe the job. A job contains one or more *documents*.

A *job set* is a set of jobs that are queued and scheduled together according to a specified scheduling algorithm for a specified device or set of devices. For implementations that embed the SNMP agent in the device, the MIB job set normally represents all the jobs known to the device. If the SNMP agent is implemented in a server that controls one or more devices, each MIB job set represents a job queue for (1) a specific device or (2) set of devices, if the server uses a single queue to load balance between several devices. Each job set is disjoint; no job shall be represented in more than one MIB job set.

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

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

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

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

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

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

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

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

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

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

An *agent* is the network entity that accepts SNMP requests from a *monitor* and implements the Job Monitoring MIB.

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

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

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

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

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

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

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

An *attribute* is a name, value-pair that specifies an instruction, a status, or a condition in a job or a document in a job submission protocol. An attribute need not be present in each job instance. In other words, attributes are present in a job instance only when there is a need to express the value. The term "attribute" will be used when discussing a *job instruction* or a *document instruction* in a job submission protocol that is not embedded in the document data. The term "attribute" will also be used for the attribute table in this MIB in which entries are present only when necessary. The term "information object" or "object" for short will be used in discussing the MIB. In other words, the server or printer accepts jobs via a job submission protocol that contains job and document attributes and the SNMP agent instruments the job by returning the equivalent, possibly transformed, job and document attributes as MIB objects in response to SNMP Get requests. The agent may also represent job and document instructions that are embedded in the document data as MIB objects, depending on implementation.

An *SNMP information object* is a name, value-pair that specifies an action, a status, or a condition in an SNMP MIB.

Job monitoring using SNMP is (1) identifying jobs within the serial streams of data being processed by the server, printer or other devices, (2) creating "rows" in the job table for each job, and (3) recording information, known by the agent, about the processing of the job in that "row".

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

2.1 Job Life Cycle

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

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

The job states are intended to be the user visible. The agent shall make these states visible in the MIB, but only for the subset of job states that the implementation has. Implementations may need to have sub-states of these user-visible states. Such implementation is *not* specified in this model, is not supported by this Job Monitoring MIB, and will vary from implementation to implementation.

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

The job model does not specify how accounting and auditing is implemented, except to require that accounting and auditing logs are separate from the job life cycle and last longer than job objects. Jobs in the **completed** state are not logs, since jobs in the **completed** state are accessible via job submission and/or job management protocol operations and are removed from these job tables after a site-settable period of time. Accounting information may be copied incrementally to the accounting logs as a job processes, may be copied while the job is in the **retained** state, or may be copied while the job is in the **completed** state, depending on implementation. The same is true for auditing logs.

421 The job model has the following states:

422 **Table 2-1: Job Object Life Cycle Summary**

State	Summary Description
1. unknown	The state of the job is not known to the agent or is unknowable, or the job is not yet created or has just been purged.
2. preProcessing	The job has been created on the server or device but the submitting client is in the process of adding additional job components and no documents have started processing. The job maybe in the process of being checked by the server/device for attributes, defaults being applied, a device being selected, etc.
3. held	The job is not yet a candidate for processing for any number of reasons. The reasons are represented as bits in the **jmJobStateReasons** object. Some reasons are used in other states to give added information about the job state. See the **JmJobStateReasonsTC** textual convention for the specification of each reason and in which states the reasons may be used.
4. pending	The job is a candidate for processing, but is not yet processing.
5. processing	The job is using one or more document transforms which include purely software processes, such as interpreting a PDL, and hardware devices.
6. needsAttention	The job 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. Usually devices indicate 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 **jmDeviceIndex** object in the Job Monitoring MIB. NOTE - Instead of the **needsAttention** job state, ISO DPA uses the multi-valued **printer-state-of-printers-assigned** job attribute, so that the state of each device that a job is using can be accurately represented. However, for the Job Monitoring MIB, the simpler approach is used of adding a single **needsAttention** job state if any device that the job is using needs attention and relying on the device MIB for more information.
7. paused	The job has been indefinitely suspended by a client issuing an operation to suspend the job so that other jobs may proceed using the same devices. The client may issue an operation to resume the

State	Summary Description
	paused job at any time, in which case the server or printer places the job in the **held** or **pending** states and the job is eventually resumed at the point where the job was paused.
8. interrupted	The job has been interrupted while **processing** by a client issuing an operation that specifies another job to be run instead of the current job. The server or printer will automatically resume the interrupted job when the interrupting job completes.
9. terminating	The job is in the process of being terminated by the server or printer, 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 **jmJobStateReasons** object shall contain the reasons that the job was terminated.
10. retained	The job is being retained by the server or printer after processing and all of the media have been successfully stacked in the output bin(s). The job (1) has completed successfully or with warnings or errors, (2) has been aborted while printing by the server/device, or (3) has been cancelled by the submitting user or operator before or during processing. The job's **jmJobStateReasons** object shall contain the reasons that the job has entered the **retained** state. While in the **retained** state, all of the job's document data (and submitted resources, if any) are retained by the server or device; thus a client could issue an operation to resubmit the job (or a copy of the job) while the job is in the **retained** state. The **retained** state is conditionally mandatory. Implementations that do *not* retain jobs after they are finished processing such that the client could request that the job be repeated (or resubmitted), need not implement the **retained** state.
11. completed	The job has (1) completed processing, (2) all of the media have been successfully stacked in the output bin(s) and (3) the server/device is keeping the job in summary form for a site-settable period for purposes of aiding operators and users to determine the disposition of users' jobs. The job (1) has completed successfully or with warnings or errors, (2) has been aborted while printing by the server/device, or (3) has been cancelled by the submitting user or operator before or during processing. The job's **jmJobStateReasons** object shall contain the reasons that the job has entered the **completed** state.

State	Summary Description
	While in the **completed** state, a job's document data (and submitted resources if any) need not be retained by the server; thus a job in the **completed** state could not be reprinted. The length of time that a job may be in this state, before transitioning to **unknown**, is implementation-dependent. However, servers that implement the **completed** job-state shall retain all of the job's Job Monitoring MIB objects, except the **jmQueueGroup** objects, so that a management application accounting program can copy them to an accounting log.

The **jmJobCurrentState** object specifies the standard job states. The legal job state transitions are shown in the state transition diagram presented in Table 2-2.

Table 2-2 - Legal Job State Transition Table

Current state											
Client operations and system-generated events	unknown[1] 1	preProcessing 2	held 3	pending 4	processing 5	needsAttention 6	paused 7	interrupted 8	terminating 9	retained 10	completed 11
CreateJob	2										
AddDocument		2	3,4	3,4	5						
CloseJob		2	3,4	4	5				9		
no CloseJob within site settable time			9								
job-submission-complete=TRUE		3,4									
job-process-after-time arrives			3,4								
ModifyJob		2	3,4	3,4	5						
PauseJob			7	7	7						
ResumeJob			7								
server dispatches job to processing				5							
job's job-state-reasons changed			3,4	3,4	5						
job's transform-state-of-transforms-assigned changed					5						
device encounters a problem that needs human intervention					6						
operator fixes problem						5					
CancelJob		9	9	9	9	9	9	9	9	10	11
Server aborts		9	9	9	9						

[1] The **unknown** state can be returned if a JSP has forwarded a job to another JSP and that JSP is no longer in contact. The **unknown** state is also used for completeness to show the job state transitions on the **CreateJob** operation.

Current state											
Client operations and system-generated events	unknown[1] 1	preProcessing 2	held 3	pending 4	processing 5	needsAttention 6	paused 7	interrupted 8	terminating 9	retained 10	completed 11
job											
job abort/cancel cleanup completes									10		
ListJobAttributes		2	3	4	5	6	7	8	9	10	11
PromoteJob			3	4							
job completes processing					10						
server purges job											1

There are two approaches that implementers may use to address the problems of the end-user using the Job Monitoring MIB:

1. The **client** also supports SNMP and the Job Monitoring MIB for status/notification to the submitting user
2. The **monitor** supports SNMP and the Job Monitoring MIB for status/notification to *any* user, including the job-submitting end user; for example, the Windows Print Manager.

The following diagram illustrates the relationships between the defined entities.

```
   +-------+     +--------+
   |monitor|     | client |
   +---#---+     +-----+--+
       #               |
  SNMP #               | DPA, TIPSI, or
       #               | other job submission
+==+===#===+==+        | protocol
|  | agent |  |        |
|  +-------+  |        |
|   printer   <--------+
|     or      | Print Job Delivery Channel
|   server    |
+=============+
```

Figure 1 - Relationship between client, printer/server, management station, and agent

```
    system   printer    asset     user          user           user
    manager  operator   manager
      O         O         O         O             O              O
     /|\       /|\       /|\       /|\           /|\            /|\
     / \       / \       / \       / \           / \            / \
      |         |         |         |             |              |
+---------+ +-------+ +-------+ +-------+   +-----------+ +-----------+
|configur-| |printer| | asset | |printer|   |   user    | |   user    |
|ator     | |manager| |manager| |browser|   |application| |application|
+---------+ +-------+ +-------+ +-------+   +-----------+ +-----------+
   ^            ^         ^         ^             |             |
   |R/W         |R/W      |R        |R      +-----------+ +-----------+
   |            |         |         |       |  spooler  | |  spooler  |
   |            |         |         |       +-----------+ +-----------+
   |            |         |         |             |             |
   |            |         |         |       +-----------+ +-----------+
   |            |         |         |       |supervisor | |supervisor |
   |            |         |         |       +-----------+ +-----------+
   |            |         |         |        ^       ^     ^       ^
   |            |         |         |        |R      |R/W  |R      |R/W
   v            v         |         |        |       |     |       |
==================================================   |   =====     |
                     |                          print|        print|
                     |SNMP                       data|         data|
  +-----+        +-------+                        PCL|          PCL|
  | MIB |<------>| agent |                 PostScript|   PostScript|
  +-----+        +-------+                       NPAP|         NPAP|
                     |unspecified                etc.|         etc.|
              +=============+  +-----------------+   |             |
              |             |--|channel/Interface|<--+             |
              |             |  +-----------------+                 |
              |   PRINTER   |                                      |
              |             |  +-----------------+                 |
              |             |--|channel/Interface|<----------------+
              +=============+  +-----------------+
```

Figure 2 - One Printer's View of the Network (extracted from RFC 1759)

3. System Configurations for the Job Monitoring MIB

This section enumerates the two configurations for which the Job Monitoring MIB is intended to be used. To simplify the pictures, the *devices* are shown as *printers*. See Goals section.

3.1 Configuration 1 - client-printer

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

1. if each **client** relinquishes control of the Print Job Delivery Channel after each job (or after a number of jobs)
2. if the printer supports more than one Print Job Delivery Channel

The job submitting **client** and/or **monitor** communicates directly with an agent that is part of the printer. The agent in the printer shall keep the job in the Job Monitoring MIB as long as the job is in the Printer, and longer in order to implement the **completed** state in which monitoring programs can copy out the accounting data from the Job Monitoring MIB.

```
      all         end-user     ######## SNMP query
   +-------+     +--------+    ---- job submission
   |monitor|     | client |
   +---#---+     +--#--+--+
       #            #  |
       # ############  |
       # #             |
+==+===#=#=+==+        |
|  | agent |  |        |
|  +-------+  |        |
|   PRINTER   <--------+
|             | Print Job Delivery Channel
|             |
+=============+
```

Figure 3 - Configuration 1 - client-printer - agent in the printer

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

1. Multiple **clients** may submit jobs to a **printer**.
2. Multiple **clients** may monitor a **printer**.
3. Multiple **monitors** may monitor a **printer**.
4. A **client** may submit jobs to multiple **printers**.
5. A **monitor** may monitor multiple **printers**.

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

In the **client-server-printer** configuration 2, the **client**(s) submit jobs to an intermediate **server** by some network connection, *not* directly to the **printer**.

The job submitting **client** and/or **monitor** communicates directly with:

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

There is no SNMP Job Monitoring MIB agent in the printer in configuration 2, at least that the client or monitor are aware. In this configuration, the agent shall return the current values of the objects in the Job Monitoring MIB both for jobs the server keeps and jobs that the server has submitted to the printer. In configuration 2, the server keeps a copy of the job during the time that the server has submitted the job to the printer. Only some time *after* the printer completes the job, shall the server remove the representation of the job from the Job Monitoring MIB in the server. The agent need not access the printer, except when a monitor queries the agent using an SNMP Get for an object in the Job Monitoring MIB. Or the agent can subscribe to the notification events that the printer generates and keep the Job Monitoring MIB update to date. The agent in the server shall keep the job in the Job Monitoring MIB as long as the job is in the Printer, and longer in order to implement the **completed** state in which monitoring programs can copy out the accounting data from the Job Monitoring MIB.

```
    all          end-user
 +-------+     +----------+     ######## SNMP query
 |monitor|     |  client  |     **** non-SNMP cntrl
 +---+---#     +---#----+-+     ---- job submission
          #        #    |
           #       #    |
            #      #    |
             #=====#=+==v==+
             | agent |     |
             +-------+     |
             |    server   |
             +----+-----+--+
          control *     |
         **********     |
         *              |
+========v====+         |
|             |         |
|             |         |
|   PRINTER   <---------+
|             | Print Job Delivery Channel
|             |
+=============+
```

Figure 4 - Configuration 2 - client-server-printer - agent in the server

The Job Monitoring MIB is designed to support the following relationships (not shown in Figure 4):

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 **monitor** communicates directly with:

1. the server using a non-SNMP protocol to monitor jobs in the server 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 keep the values of the objects in the Job Monitoring MIB that the agent implements updated for a job that the server has submitted to the printer. The agent shall obtain information about the jobs submitted to the printer from the server (either in the job submission protocol, in the document data, or by direct query of the server), in order to populate some of the objects the Job Monitoring MIB in the printer. The agent in the printer shall keep the job in the Job Monitoring MIB as long as the job is in the Printer, and longer in order to implement the **completed** state in which monitoring programs can copy out the accounting data from the Job Monitoring MIB.

```
    all          end-user
 +-------+     +----------+
 |monitor|     |  client  |      ######## SNMP query
 +---+---*     +---*----+-+      **** non-SNMP query
     #    *        *    |        ---- job submission
     #     *       *    |
     #      *      *    |
     #       *=====+=+==v==+
     #       |             |
     #       +             |
     #       |    server   |
     #       +----#-----+--+
     #    optional#     |
     #   ##########     |
     #   #              |
+==+=v===v=+==+         |
|  | agent |  |         |
|  +-------+  |         |
|   PRINTER   <---------+
|             | Print Job Delivery Channel
|             |
+=============+
```

Figure 5 - Configuration 3 - client-server-printer - client monitors printer agent and server

The Job Monitoring MIB is designed to support the following relationships (not shown in Figure 5):

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**.

4. Conformance Considerations

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

4.1 Conformance Terminology

This specification uses the verbs: "*shall*", "*should*", "*may*", and "*need not*" to specify conformance requirements as follows:

- "shall": indicates an action that the subject of the sentence must implement in order to claim conformance to this specification
- "may": indicates an action that the subject of the sentence does not have to implement in order to claim conformance to this specification, in other words that action is an implementation option
- "need not": indicates an action that the subject of the sentence does not have to implement in order to claim conformance to this specification. The verb "need not" is used instead of "may not", since "may not" sounds like a prohibition.
- "should": indicates an action that is recommended for the subject of the sentence to implement, but is not required, in order to claim conformance to this specification.

4.2 Agent Conformance Requirements

An agent shall implement all mandatory groups in this specification. An agent shall implement conditionally mandatory groups, if the server or device that the agent is instrumenting has the features represented by the objects in the conditionally mandatory group. This section also lists the objects from other IETF MIB specifications that are mandatory for conformance by an agent to this Job Monitoring MIB specification.

4.2.1 MIB II System Group objects

The Job Monitoring MIB agent shall implement all objects in the system group of MIB-II (RFC 1213), whether the Printer MIB is implemented or not.

4.2.2 MIB II Interface Group objects

The Job Monitoring MIB agent shall implement all objects in the Interfaces Group of MIB-II (RFC 1213), whether the Printer MIB is implemented or not.

4.2.3 Printer MIB objects

If the agent is instrumenting a device that is a printer, the agent shall implement all of the mandatory objects in the Printer MIB and all the objects in other MIBs that conformance to the Printer MIB requires, such as the Host Resources MIB. If the agent is instrumenting a server that controls one or more networked printers, the agent need not implement the Printer MIB and need not implement the Host Resources MIB.

4.3 Job Monitoring Application Conformance Requirements

A job monitoring application (monitor) is a management or client application that uses SNMP to access the agent that implements this Job Monitoring MIB. A job monitoring application shall accept all objects in all mandatory and conditionally mandatory groups that are required to be implemented by an agent according to Section 4.2 and shall either present them to the user or ignore them.

A job monitoring application shall accept all enum values and bit vector bits specified in this standard and additional ones that may be registered with IANA and shall either present them to the user or ignore them. See Section 7 entitled "IANA Considerations" on page 29.

5. Job Identification

The purpose of the Job Identification objects is to allow the user, operator, or the system administrator to identify the jobs of interest. The Job Monitoring MIB needs to provide for identification of the job at both sides of the job submission process. The primary identification point must be at the client side. The client side identifiers allow the user to identify the job of interest from all the jobs currently "known" by the server or device. The client side identifiers can be assigned by either the client's local system or a downstream server or device. The point of assignment will be determined by the job submission protocol in use. Two client-side objects are provided: **jmJobIdName** and **jmJobIdNumber** so that both textual identifiers and numeric identifiers can be represented, depending on the job submission protocol. The intent is that the agent shall provide the same values for these two client-side objects as the user is provided for by the job submission protocol that happens to be in use. The client-side job identifiers in combination should provide the user and operator with unique job identifications.

The server/device-side identifier will be assigned by the server or device that accepts the jobs from submitting clients. The MIB agent shall use the job identifier assigned by the server or device to the job as the value of the **jmJobIndex** object that defines the table rows (there are multiple tables) that contain the information relating to the job. This object allows the interested party to obtain all objects desired that relate to this job.

The **jmJobName** object provides a name that the user supplies an a job attribute with the job. It is not necessarily unique, even for one user, let alone across users.

6. Internationalization Considerations

There are a number of objects in this MIB that are represented as coded character sets. The data type for such objects is **OCTET STRING**. See Section 12 entitled "Datatypes used in the Job Monitoring MIB" on page 32. Such objects could be in different coded character sets and could be localized in the language and country, i.e., could be localized.

However, for the Job Monitoring MIB, most of the objects are supplied as job attributes by the client that submits the job to the server or device and so are represented in the coded character set specified by that client. Therefore, the agent is *not* able to provide for different representations depending on the locale of the server, device, or user of the job monitoring application. The only exception is job submission protocols that pass job or document attributes as OBJECT IDENTIFIERS or enums. For those job and document attributes, the agent shall represent the corresponding objects in the Job Monitoring MIB as coded character sets in the current (default) locale of the server or printer as established by the system administrator or the implementation.

For simplicity, this specification assumes that the clients, job monitoring applications, servers, and devices are all running in the same locale. However, this specification allows them to run in any locale, including locales that use two-octet coded character sets, such as ISO 10646 (Unicode). Job monitors applications are expected to understand the coded character set of the client (and job), server, or device. No special means is provided for the monitor to discover the coded character set used by jobs or by the server or device. This specification does *not* contain an object that indicates what locale the server or device is running in, let alone contain an object to control what locale the agent is to use to represent coded character set objects.

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

7. IANA Considerations

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

7.1 IANA Registration of enums

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

This working group has defined several type of enumerations for use in the Job Monitoring MIB and the Printer MIB (see RFC 1759). These enumerations differ in the method employed to control the addition of new enumerations. Throughout this document, references to "type n enum", where n can be 1, 2 or 3 can be found in the various tables. The definitions of these types of enumerations are:

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

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

Type 2 enumeration: An initial set of values are defined in the Job Monitoring MIB specification. Additional enumerated values are registered after review by this working group. The initial versions of the MIB will contain the values registered so far. After the MIB is approved, additional values will be registered through IANA after approval by this working group.

The following type 2 enums are contained in the current draft (see table of contents Table of Textual-Conventions):

1. **JmJobServiceTypesTC**
2. **JmJobStateTC**
3. **JmAttributeTypeTC**

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

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

7.2 IANA Registration of bit string values

This draft contains the following bit string textual-conventions:

1. **JmJobStateReasonsTC**

The **jmJobStateReasons** object is defined as a bit string using the **JmJobStateReasonsTC** textual-convention that is represented by an **OCTET STRING(SIZE(0..63))**. Bits in the bit string are assigned starting with the most significant bit in the most significant octet which is called bit 1. Bit 2 is the next most significant bit in the most significant octet, etc. Bit 9 is the most significant bit in the second most significant octet, etc., up to the maximum bit: 504 (= 8 x 63). The registration of **JmJobStateReasonsTC** bit values shall follow the procedures for a type 2 enum as specified in Section 7.1

8. Security Considerations

8.1 Read-Write objects

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

8.2 Read-Only Objects In Other User's Jobs

The security policy of some sites may be that unprivileged users can only get the objects from jobs that they submitted, plus a few minimal objects from other jobs, such as the **jobKOctetsTotal** and **jobKOctetsCompleted** attributes**,** so that a user can tell how busy a printer is. Other sites might allow all unprivileged users to see all objects of all jobs. It is up to the agent to implement any such restrictions based on the identification of the user making the SNMP request. This MIB does not require, nor does it specify how, such restrictions would be implemented.

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

9. Returning Objects With No Value In Mandatory Groups

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

10. Notification and Traps

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

11. Object Groups and Tables

There is a one to one relationship between tables and groups as follows:

Group	**Table**	**Description**	**No. of accessible objects**	**Conformance**
jmGeneralGroup	N/A	General information about a job set (queue).	5	Mandatory
jmQueueGroup	**jmQueueTable**	Ordered list of jobs that have *not* finished and job information that relevant only until the job has finished processing. Mandatory only if queuing (or spooling).	6	Conditionally mandatory
jmCompletedGroup	**jmCompletedTable**	Ordered list of pointers to jobs that have finished processing.	3	Mandatory
jmJobGroup	**jmJobTable**	Basic job identification and	9	Mand

Group	Table	Description	No. of accessible objects	Conformance
		status information.		atory
jmAttributeGroup	**jmAttributeTable**	Attributes representing (1) job and document information, (2) resources required, and (3) resources consumed by the job. Can have more than one attribute of the same type per job.	4	Mandatory
		Mandatory Totals:	21	
		Conditionally Mandatory Totals:	6	
		Totals:	27	

12. Datatypes used in the Job Monitoring MIB

The following datatypes are used in the Job Monitoring MIB

Table 12-1 - MIB Datatype specifications

OCTET STRING(SIZE(0..63))	**Octet String** 0 to 63 octets with 63 octets maximum length). See ISO/ITU Abstract Syntax and Notation (ASN.1), ISO/ITU 8824/X.208. The OCTET STRING is used for the following purposes: 1. Sequence of arbitrary binary data 2. Sequence of one- or two-octet character coded data. This character coded data is supplied by the client that submits the job to the server or printer/device and so is in the coded character set specified by that client. In some job submission protocols, some job and document attributes are represented as enumerations or OBJECT IDENTIFIERS by the client. In such cases the Job Monitoring MIB agent shall represent the objects of type OCTET STRING in the coded character set established by the system administrator or implementer of the server or printer/device. Monitors are expected to understand the coded character set of the client (and job), server, or printer/device. No special means is provided for the monitor to discover the coded character set used by jobs or by the server or printer/device. A zero length string is a valid value that a submitting user and/or a

	receiving job submission server/device might assign to a job attribute. If a job attribute of type OCTET STRING does not have any value, either (1) because the submitting user or client did not supply a value and the recipient server or printer/device did not assign a default value or (2) because the job submission protocol does not support that job attribute, the agent shall return a zero-length string. See Section 9 Returning Objects With No Value In Mandatory Groups on page 31 3. Bit string. Bits are assigned and numbered starting at 1 for the most significant bit of the most significant octet. IANA handles registration of bits assigned after this standard is approved. See Section 7 entitled IANA Considerations on page 29.
Integer32	32-bit **Integer** with explicit range indicated - for unsigned quantities, the range is specified as 0..2147483647 (2^31-1) or 1..2147483647 to avoid using the sign bit which avoids implementation problems with signed vs. unsigned representation. See IETF SNMPv2-SMI (RFC 1902).
Counter32	32-bit unsigned counter. See IETF SNMPv2-SMI (RFC 1902).
DateAndTime	**DateAndTime** from SMIv2 textual-conventions, RFC 1903 and later. An 8 or 11 octet string with each octet or pair of octets coded as binary integers that contain the year(2), month(1), day(1), hour(1), minute(1), second(1), deci-seconds(1) and, optionally, the direction (+/-), hours(1), and minutes(1) from UTC. See SMIv2-TC (RFC 1903) for details. NOTE: **DateAndTime** is *not* a printable string of coded characters.
TimeStamp	Time kept in hundredths of a second: the value of MIB-II's sysUpTime object when an event (epoch) occurred. See SMIv2-TC (RFC 1903) for details.
XxxYyyZzzzTC	Textual Convention for specifying enums. The following specification for enumerations has been adapted from the Printer MIB, RFC 1759: Enumerations (enums) are sets of symbolic values defined for use with one or more objects. All enumeration sets are assigned a symbolic data type name (textual convention). These enumerations are listed at the beginning of this specification. See Section 7 entitled IANA Considerations on page 29.

13. MIB specification

The following pages constitute the actual Job Monitoring MIB.

```
Job-Monitoring-MIB DEFINITIONS ::= BEGIN

IMPORTS
    MODULE-IDENTITY, OBJECT-TYPE, experimental,
    Integer32                                        FROM SNMPv2-SMI
    TEXTUAL-CONVENTION, DateAndTime                  FROM SNMPv2-TC
    MODULE-COMPLIANCE, OBJECT-GROUP                  FROM SNMPv2-CONF;


-- Use the experimental (54) OID assigned to the Printer MIB before it
-- was published as RFC 1759.
-- Upon publication of the Job Monitoring MIB as an RFC, delete this
-- comment and the line following this comment and change the
-- reference of { temp 104 } (below) to { mib-2 X }.
-- This will result in changing:
-- 1 3 6 1 3 54 jobmonmib(105)    to:
-- 1 3 6 1 2  1 jobmonmib(X)
-- This will make it easier to translate prototypes to
-- the standard namespace because the lengths of the OIDs won't
-- change.
temp OBJECT IDENTIFIER ::= { experimental 54 }

jobmonmib MODULE-IDENTITY
    LAST-UPDATED "9703260000Z"
    ORGANIZATION "IETF Printer MIB Working Group"
    CONTACT-INFO
        "Tom Hastings
        Postal:  Xerox Corp.
                 Mail stop ESAE-231
                 701 S. Aviation Blvd.
                 El Segundo, CA 90245

        Tel:     (301)333-6413
        Fax:     (301)333-5514
        E-mail:  hastings@cp10.es.xerox.com"
    DESCRIPTION
        "The MIB module for monitoring job in servers, printers, and
        other devices.

        File: jmp-mib.doc,  .pdf,  .txt,  .mib
        Version: 0.71"
    ::= { temp 105 }
```

```
-- Textual conventions for this MIB module
```

-- textual-convention 1: JmJob Service**TypesTC**

```
JmJobServiceTypesTC ::= TEXTUAL-CONVENTION
    STATUS      current
    DESCRIPTION
        "Specifies the type(s) of service to which the job has been
        submitted (print, fax, scan, etc.).  The service type is
        represented as an enum that 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 jmJobServiceTypes object,
        corresponding to the values: 8+32+4=44, respectively.

        Whether this object 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.  With
        either implementation, the agent shall return a non-zero value
        for this object indicating the type of the job.

        One of the purposes of this object 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 the object is in the job identification
        category.

        The following service component types are defined and are
        assigned a separate bit value in the enum for use with the
        jmJobServiceTypes object:"

    -- This is a type 2 enumeration.  See Section 7.1 on page 29.
    SYNTAX      INTEGER {
        other(1),       -- The job contains some document production
                        -- instructions that are not one of the
                        -- identified types.

        unknown(2),     -- The job contains some document production
                        -- instructions whose type is unknown to the
                        -- agent.

        print(4),       -- The job contains some document production
                        -- instructions that specify printing

        scan(8),        -- The job contains some document production
                        -- instructions that specify scanning

        faxIn(16),      -- The job contains some document production
```

```
                              -- instructions that specify receive fax

          faxOut(32),         -- The job contains some document production
                              -- instructions that specify sending fax

          getFile(64),        -- The job contains some document production
                              -- instructions that specify accessing files or
                              -- documents

          putFile(128),       -- The job contains some document production
                              -- instructions that specify storing files or
                              -- documents

          mailList(256)       -- The job contains some document production
                              -- instructions that specify distribution of
                              -- documents using an electronic mail system.

      }
```

-- textual-convention 2: JmJobStateTC

```
JmJobStateTC ::= TEXTUAL-CONVENTION
    STATUS      current
    DESCRIPTION
        "The current state of the job (pending, processing, held, etc.)

        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.

        A companion textual convention (JmJobStateReasonsTC) and
        corresponding object (jmJobStateReasons) provide additional
        information about job states.  While the job states cannot be
        added to without impacting deployed clients, it is the intent
        that additional JmJobStateReasonsTC enums can be defined without
        impacting deployed clients.  In other words, the
        JmJobStateReasonsTC is intended to be extensible.  See page 42.

        The following job state standard values are defined by adding
        (+2) to the last arc of the ISO DPA OBJECT IDENTIFIER value of
        the job-current-state job attribute:"

    -- This is a type 2 enumeration.  See Section 7.1 on page 29.
    SYNTAX      INTEGER {
        other(1),           -- The job state is not one of the defined
                            -- states.

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

        preProcessing(3),   -- The job has been created on the server or
                            -- device but the submitting client is in
                            -- the process of adding additional job
                            -- components and no documents have started
                            -- processing.  The job maybe in the process
                            -- of being checked by the server/device for
                            -- attributes, defaults being applied, a
                            -- device being selected, etc.

        held(12),           -- The job is not yet a candidate for
                            -- processing for any number of reasons.
                            -- The reasons are represented as bits in
                            -- the jmJobStateReasons object.  Some
                            -- reasons are used in other states to give
                            -- added information about the job state.
                            -- See the JmJobStateReasonsTC textual
                            -- convention for the specification of each
                            -- reason and in which states the reasons
                            -- may be used.
       pending(6),          -- The job is a candidate for processing,
                            -- but is not yet processing.

       processing(7),       -- The job is using one or more document
                            -- transforms which include purely software
                            -- processes, such as interpreting a PDL,
                            -- and hardware devices.

       needsAttention(9),   -- The job 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.
                            --
                            -- Usually devices indicate 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 jmDeviceIndex
                            -- object in the Job Monitoring MIB.
                            --
                            -- NOTE - Instead of the needsAttention job
                            -- state, ISO DPA uses the multi-valued
                            -- printer-state-of-printers-assigned job
                            -- attribute, so that the state of each
                            -- device that a job is using can be
                            -- accurately represented.  However, for the
                            -- Job Monitoring MIB, the simpler approach
                            -- is used of adding a single needsAttention
                            -- job state if any device that the job is
                            -- using needs attention and relying on the
                            -- device MIB for more information.

       paused(13),          -- The job has been indefinitely suspended
                            -- by a client issuing an operation to
                            -- suspend the job so that other jobs may
                            -- proceed using the same devices.  The
                            -- client may issue an operation to resume
                            -- the paused job at any time, in which case
                            -- the server or printer places the job in
                            -- the held or pending states and the job is
                            -- eventually resumed at the point where the
                            -- job was paused.

       interrupted(8),      -- The job has been interrupted while
                            -- processing by a client issuing an
                            -- operation that specifies another job to
                            -- be run instead of the current job.  The
                            -- server or printer will automatically
                            -- resume the interrupted job when the
```

```
                        -- interrupting job completes.

terminating(14),        -- The job is in the process of being
                        -- terminated by the server or printer,
                        -- 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
                        -- jmJobStateReasons object shall contain
                        -- the reasons that the job was terminated.

retained(11),           -- The job is being retained by the server
                        -- or printer after processing and all of
                        -- the media have been successfully stacked
                        -- in the output bin(s).
                        --
                        -- The job (1) has completed successfully or
                        -- with warnings or errors, (2) has been
                        -- aborted while printing by the
                        -- server/device, or (3) has been cancelled
                        -- by the submitting user or operator before
                        -- or during processing.  The job's
                        -- jmJobStateReasons object shall contain
                        -- the reasons that the job has entered the
                        -- retained state.
                        --
                        -- While in the retained state, all of the
                        -- job's document data (and submitted
                        -- resources, such as fonts, logos, and
                        -- forms, if any) are retained by the server
                        -- or device; thus a client could issue an
                        -- operation to resubmit the job (or a copy
                        -- of the job) while the job is in the
                        -- retained state.
                        --
                        -- The retained state is conditionally
                        -- mandatory.  Implementations that do not
                        -- retain jobs after they are finished
                        -- processing such that the client could
                        -- request that the job be repeated (or
                        -- resubmitted), need not implement the
                        -- retained state.

completed(17)           -- The job has (1) completed after
                        -- processing and all of the media have been
                        -- successfully stacked in the output bin(s)
                        -- and (2) the server/device is keeping the
                        -- job in summary form for a site-settable
                        -- period for purposes of aiding operators
                        -- and users to determine the disposition of
                        -- users' jobs.
                        --
                        -- The job (1) has completed successfully or
```

```
                                          -- with warnings or errors, (2) has been
                                          -- aborted while printing by the
                                          -- server/device, or (3) has been cancelled
                                          -- by the submitting user or operator before
                                          -- or during processing.  The job's
                                          -- jmJobStateReasons object shall contain
                                          -- the reasons that the job has entered the
                                          -- completed state.
                                          --
                                          -- While in the completed state, a job's
                                          -- document data (and submitted resources,
                                          -- such as fonts, logos, and forms, if any)
                                          -- need not be retained by the server; thus
                                          -- a job in the completed state could not be
                                          -- reprinted. The length of time that a job
                                          -- may be in this state, before
                                          -- transitioning to unknown, is
                                          -- implementation-dependent.  However,
                                          -- servers that implement the completed job-
                                          -- state shall retain all of the job's Job
                                          -- Monitoring MIB objects, except the
                                          -- jmQueueGroup objects, so that a
                                          -- management application accounting program
                                          -- can copy them to an accounting log.
907       }
```

-- textual-convention 3: JmJobStateReasonsTC

```
JmJobStateReasonsTC ::= TEXTUAL-CONVENTION
    STATUS      current
    DESCRIPTION
        "This textual-convention is used in the jmJobStateReasons object
        to provides additional information regarding the
        jmJobCurrentState object.  The jmJobStateReasons object
        identifies the reason or reasons that the job is in the
        preProcessing, held, pending, processing, needsAttention,
        paused, interrupted, terminating, retained, or completed state.
        The server shall indicate the particular reason(s) by setting
        the value of the jmJobStateReasons object.  While the job states
        cannot be added to without impacting deployed clients, it is the
        intent that additional JmJobStateReasonsTC enums can be defined
        without impacting deployed clients.  In other words, the
        JmJobStateReasonsTC is intended to be extensible.

        When the job does not have any reasons for being in its current
        state, the server shall set the value of the jmJobStateReasons
        object to a bit string containing all zeros.

        Bits in the bit string are assigned starting with the most
        significant bit in the most significant octet which is called
        bit 1.  Bit 2 is the next most significant bit in the most
        significant octet, etc.  Bit 9 is the most significant bit in
        the second most significant octet, etc., up to the maximum bit:
        504 (= 8 x 63).

        An agent need only return the most significant octet up to the
        least significant octet that contains a non-zero bit.

        If all bits are zero, the agent may return an OCTET STRING of
        zero length.  Alternatively, an agent may always return a fixed
        number of octets starting with the most significant octet and
        running through the least significant octet that could ever have
        a one bit in it for that implementation.

        This object is a type 2 bit string.  See Section 7 entitled
        'IANA Considerations' on page 29 and Section 0 entitled
        'Datatypes used in the Job Monitoring MIB' on page 32.

        The following standard values are defined as bit numbers, not
        enums (the bit number equals the last arc of DPA id-val-reasons-
        xxx OID for the reasons that are in ISO DPA):"

    -- This is a type 2 bit string.  See section 7.2 on page 30.
    SYNTAX      INTEGER {
    --   really OCTET STRING(SIZE(0..63))
        documentsNeeded(1),    -- The job is in the held state because
                               -- the server or printer is waiting for
```

```
                      -- the job's files to start and/or finish
                      -- being transferred before the job can
                      -- be scheduled to be printed.

jobHoldSet(2),        -- The job is in the held state because
                      -- the client specified that the job is
                      -- to be held.

jobProcessAfterSpeci  -- The job is in the held state because
fied(3),              -- the client specified a time
                      -- specification reflected in the value
                      -- of the job's
                      -- jmJobProcessAfterDateAndTime object
                      -- that has not yet occurred.

requiredResourcesNot  -- The job is in the held state because
Ready(4),             -- at least one of the resources needed
                      -- by the job, such as media, fonts,
                      -- resource objects, etc., is not ready
                      -- on any of the physical devices for
                      -- which the job is a candidate.

successfulCompletion  -- The job is in the retained or
(5),                  -- completed state having completed
                      -- successfully.

completedWithWarning  -- The job is in the terminating,
s(6),                 -- retained, or completed states having
                      -- completed with warnings.

completedWithErrors(  -- The job is in the terminating,
7),                   -- retained, or completed states having
                      -- completed with errors (and possibly
                      -- warnings too).

cancelledByUser(8),   -- The job is in the terminating,
                      -- retained, or completed states having
                      -- been cancelled by the user.

cancelledByOperator(  -- The job is in the terminating,
9),                   -- retained, or completed states having
                      -- been cancelled by the operator using
                      -- the CancelJob request.

abortedBySystem(10),  -- The job is in the terminating,
                      -- retained, or completed states having
                      -- been aborted by the system.

logfilePending(11),   -- The job's logfile is pending file
                      -- transfer.

logfileTransferring(  -- The job is in the terminating,
12),                  -- retained, or completed states and the
```

```
                        -- job's logfile is being transferred.

cascaded(13),           -- After the outbound gateway retrieves
                        -- all job and document attributes and
                        -- data, it stores the information into a
                        -- spool directory.  Once it has done
                        -- this, it sends the supervisor a job-
                        -- processing event with this job-state-
                        -- reason which tells the supervisor to
                        -- transition to a new job state.

deletedByAdministrat    -- The administrator has issued a Delete
or(14),                 -- operation on the job or a Clean
                        -- operation on the server or queue
                        -- containing the job; therefore the job
                        -- may have been cancelled before or
                        -- during processing, and will have no
                        -- retention-period or completion-period.

discardTimeArrived(1    -- The job has been deleted (cancelled
5),                     -- with the job-retention-period set to
                        -- 0) due to the fact that the time
                        -- specified by the job's job-discard-
                        -- time has arrived [if the job had
                        -- already completed, the only action
                        -- that would have occurred is that the
                        -- job-retention-period would be set to 0
                        -- and the job is deleted].

postProcessingFailed    -- The post-processing agent failed while
(16),                   -- trying to log accounting attributes
                        -- for the job; therefore the job has
                        -- been placed into retained state for a
                        -- system-defined period of time, so the
                        -- administrator can examine it, resubmit
                        -- it, etc.  The post-processing agent is
                        -- a plug-and-play mechanism which the
                        -- system and the customer uses to add
                        -- functionality that is executed after a
                        -- job has finished processing.

submissionInterrupte    -- Indicates that the job was not
d(17),                  -- completely submitted for the following
                        -- reasons: (1) the server has crashed
                        -- before the job was closed by the
                        -- client.  The server shall put the job
                        -- into the completed state (and shall
                        -- not print the job). (2) the server or
                        -- the document transfer method has
                        -- crashed in some non-recoverable way
                        -- before the document data was entirely
                        -- transferred to the server.  The server
                        -- shall put the job into the completed
                        -- state (and shall not print the job).
                        -- (3) the client crashed or failed to
                        -- close the job before the time-out
                        -- period.  The server shall close the
                        -- job and put the job into the held
                        -- state with job-state-reasons of
                        -- submission-interrupted and job-hold-
                        -- set and with the job's job-hold
                        -- attribute set to TRUE.  The user may
                        -- release the job for scheduling by
                        -- issuing a job submission or management
                        -- protocol operation.

maxJobFaultCountExce    -- The job has been faulted and returned
eded(18),               -- by the server several times and that
                        -- the job-fault-count exceeded the
                        -- device's (or server's, if not defined
                        -- for the device) cfg-max-job-fault-
                        -- count.  The job is automatically put
                        -- into the held state regardless of the
                        -- hold-jobs-interrupted-by-device-
                        -- failure attribute. This job-state-
                        -- reasons value is used in conjunction
                        -- with the job-interrupted-by-device-
                        -- failure value.

devicesNeedAttention    -- One or more document transforms that
TimeOut(19),            -- the job is using needs human
                        -- intervention in order for the job to
                        -- make progress, but the human
                        -- intervention did not occur within the
                        -- site-settable time-out value and the
                        -- server/device has transitioned the job
                        -- to the held state.

needsKeyOperatorTime    -- One or more devices or document
Out(20),                -- transforms that the job is using need
                        -- a specially trained operator (who may
                        -- need a key to unlock the device and
                        -- gain access) in order for the job to
                        -- make progress, but the key operator
                        -- intervention did not occur within the
                        -- site-settable time-out value and the
                        -- server/device has transitioned the job
                        -- to the held state.

jobStartWaitTimeOut(    -- The server/device has stopped the job
21),                    -- at the beginning of processing to
                        -- await human action, such as installing
                        -- a special cartridge or special non-
                        -- standard media, but the job was not
                        -- resumed within the site-settable time-
                        -- out value and the server/device has
```

```
                            -- transitioned the job to the held
                            -- state.  Normally, the job is resumed
                            -- by means outside the job submission
                            -- protocol, such as some local function
                            -- on the device.

jobEndWaitTimeOut(22        -- The server/device has stopped the job
),                          -- at the end of processing to await
                            -- human action, such as removing a
                            -- special cartridge or restoring
                            -- standard media, but the job was not
                            -- resumed within the site-settable time-
                            -- out value and the server/device has
                            -- transitioned the job to the retained
                            -- state.  Normally, the job is resumed
                            -- by means outside the job submission
                            -- protocol, such as some local function
                            -- on the device, whereupon the job shall
                            -- transition immediately to the
                            -- terminating state.

jobPasswordWaitTimeO        -- The server/device has stopped the job
ut(23),                     -- at the beginning of processing to
                            -- await input of the job's password, but
                            -- the human intervention did not occur
                            -- within the site-settable time-out
                            -- value and the server/device has
                            -- transitioned the job to the held
                            -- state.  Normally, the password is
                            -- input and the job is resumed by means
                            -- outside the job submission protocol,
                            -- such as some local function on the
                            -- device.

deviceTimedOut(24),         -- A device that the job was using has
                            -- not responded in a period specified by
                            -- the device's site-settable attribute.

connectingToDeviceTi        -- The server is attempting to connect to
meOut(25),                  -- one or more devices which may be dial-
                            -- up, polled, or queued, and so may be
                            -- busy with traffic from other systems,
                            -- but server was unable to connect to
                            -- the device within the site-settable
                            -- time-out value and the server has
                            -- transitioned the job to the held
                            -- state.

transferring(26),           -- The job is being transferred to a down
                            -- stream server or device.

queuedInDevice(27),         -- The job has been queued in a down
                            -- stream server or device.
```

```
jobCleanup(28),       -- The server/device is performing
                      -- cleanup activity as part of ending
                      -- normal processing.

processingToStopPoin  -- The requester has issued an operation
t(29),                -- to interrupt the job and the
                      -- server/device is processing up until
                      -- the specified stop point occurs.

jobPasswordWait(30),  -- The server/device has selected the job
                      -- to be next to process, but instead of
                      -- assigning resources and started the
                      -- job processing, the server/device has
                      -- transitioned the job to the held state
                      -- to await entry of a password (and
                      -- dispatched another job, if there is
                      -- one).  The user resumes the job either
                      -- locally or by issuing a remote
                      -- operation and supplying a job-
                      -- password=secret-code input parameter
                      -- that must match the job's job-password
                      -- attribute.

validating(31),       -- The server/device is validating the
                      -- job after a CreateJob operation.  The
                      -- job state may be creating, held,
                      -- pending, or processing.

queueHeld(32),        -- The operator has held the entire queue
                      -- by means outside the scope of the Job
                      -- model.

jobProofWait(33),     -- The job has produced a single proof
                      -- copy and is in the held state waiting
                      -- for the requester to issue an
                      -- operation to release the job to print
                      -- normally, obeying the job-copies and
                      -- copy-count job and document attributes
                      -- that were originally submitted.

heldForDiagnostics(3  -- The system is running intrusive
4),                   -- diagnostics, so the all jobs are being
                      -- held.

serviceOffLine(35),   -- The service/document transform is off-
                      -- line and accepting no jobs.  All
                      -- pending jobs are put into the held
                      -- state.  This could be true if its
                      -- input is impaired or broken.

noSpaceOnServer(36),  -- The job is held because there is no
                      -- room on the server to store all of the
```

```
                          -- job.  For example, there is no room
                          -- for the document data or a scan-to-
                          -- file job.

    pinRequired(37),      -- The System Administrator settable
                          -- device policy is (1) to require PINs,
                          -- and (2) to hold jobs that do not have
                          -- a pin supplied as an input parameter
                          -- when the job was created. The
                          -- requester shall either (1) enter a pin
                          -- locally at the device or issue a
                          -- remote operation supplying the PIN in
                          -- order for the job to be able to
                          -- proceed.

    exceededAccountLimit  -- The account for which this job is
    (38),                 -- drawn has exceeded its limit.  This
                          -- condition should be detected before
                          -- the job is scheduled so that the user
                          -- does not wait until his/her job is
                          -- scheduled only to find that the
                          -- account is overdrawn.  This condition
                          -- may also occur while the job is
                          -- processing either as processing begins
                          -- or part way through processing.
                          --
                          -- An overdraft mechanism should be
                          -- included to be user-friendly, so as to
                          -- minimize the chances that the job
                          -- cannot finish or that media is wasted.
                          -- For example, the server/device should
                          -- finish the current copy for a job with
                          -- collated document copies, rather than
                          -- stopping in the middle of the current
                          -- document copy.

    heldForRetry(39),     -- The job encountered some errors that
                          -- the server/device could not recover
                          -- from with its normal retry procedures,
                          -- but the error is worth trying the job
                          -- later, such as phone number busy or
                          -- remote file system in-accessible.  For
                          -- such a situation, the server/device
                          -- shall add the held-for-retry value to
                          -- the job's jmJobStateReasons object and
                          -- transition the job from the processing
                          -- to the held, rather than to the
                          -- retained state.


-- The following values are from the X/Open PSIS draft standard:

                          -- The job was cancelled because the
```

```
                             -- server or device was shutdown before
                             -- completing the job.  The job shall be
                             -- placed in the pending state [if the
        cancelledByShutdown( -- job was not started, else the job
        40),                 -- shall be placed in the terminating
                             -- state].

        deviceUnavailable(41 -- This job was aborted by the system
        ),                   -- because the device is currently unable
                             -- to accept jobs. This reason [shall be]
                             -- used in conjunction with the reason
                             -- aborted-by-system. The job shall be
                             -- placed in the pending state.

        wrongDevice(42),     -- This job was aborted by the system
                             -- because the device is unable to handle
                             -- this particular job; the spooler
                             -- should try another device.  This
                             -- reason [shall be] used in conjunction
                             -- with the reason aborted-by- system.
                             -- The job shall be pending if the queue
                             -- contains other physical devices that
                             -- the job could print on, and the
                             -- spooler is capable of not sending the
                             -- job back to a physical device that has
                             -- rejected the job for this job-state-
                             -- reasons value. Otherwise, [the job]
                             -- shall be retained.

        badJob(43),          -- This job was aborted by the system
                             -- because this job has a major problem,
                             -- such as an ill-formed PDL; the spooler
                             -- should not even try another device.
                             -- This reason shall be used in
                             -- conjunction with the reason aborted-
                             -- by-system. The job shall be placed in
                             -- the terminating state.

        jobInterruptedByDevi -- A device or the print system software
        ceFailure(44),       -- that the job was using has failed
                             -- while the job was processing.  The
                             -- device is keeping the job in the held
                             -- state until an operator can determine
                             -- what to do with the job.


-- The following additional job state reasons have been added to align
-- with the Internet Printing Protocol (IPP):

        jobPrinting(45)      -- The job is putting marks on a medium.
                             -- This optional job state reason is
                             -- provided for systems where there is a
                             -- significant difference in the time
```

```
                                      -- period while a job is in the
                                      -- processing state between putting marks
                                      -- on a medium and other activities, such
                                      -- as interpreting the document data.
                                      -- For systems that interpret and mark at
                                      -- the same time for a job need not
                                      -- implement this job state reason.
        }
```

--   The following table shows the **JmJobStateReasonsTC** values and the
--   job states for which they are applicable. The ISO DPA job state
--   reasons are shown along with additional job-state-reasons that
--   give users additional feedback on the progress of their job:
--

--		Job States							
-- -- --		held	pending	processing	paused	interrupted	terminating	retained	completed
--	Descriptive Name	ISO DPA values							
--	documents-needed(1)	x							
--	job-hold-set(2)	x							
-- --	job-process-after-specified(3)	x							
-- --	required-resources-not-ready(4)	x							
-- --	successful-completion(5)							x	x
-- --	completed-with-warnings(6)							x	x
-- --	completed-with-errors(7)							x	x
-- --	cancelled-by-user(8)						x	x	x
-- --	cancelled-by-operator(9)						x	x	x
-- --	aborted-by-system(10)						x	x	x
--	logfile-pending(11)						x	x	
-- --	logfile-transferring(12)						x	x	
--		Additional reasons							
-- -- --	Descriptive Name	held	pending	processing	paused	interrupted	terminating	retained	completed
--	cascaded(13)						x	x	x
-- --	deleted-by-administrator(14)						x	x	x
-- --	discard-time-arrived(15)						x	x	x
-- --	postprint-failed(16)						x	x	x
-- --	submission-interrupted(17)						x	x	x
-- --	max-job-fault-count-exceeded(18)						x	x	x
-- -- --	devices-need-attention-time-out(19)	x					x	x	x

--		Job States							
-- -- --		held	pending	processing	paused	interrupted	terminating	retained	completed
--	Descriptive Name	ISO DPA values							
-- --	needs-key-operator-time-out(20)	x					x	x	x
-- --	job-start-wait-time-out(21)	x					x	x	x
-- --	job-end-wait-time-out(22)						x	x	x
-- --	job-password-wait-time-out(23)	x	x						
-- --	device-timed-out(24)	x					x	x	x
-- --	connecting-to-device-time-out(25)	x					x	x	x
--	transferring(26)			x					
-- --	queued-in-device(27)			x					
--	job-cleanup(28)			x					
-- --	processing-to-stop-point(29)			x					
-- --	job-password-wait(30)	x		x					
--	validating(31)	x	x	x					
--	queue-held(32)	x							
--	job-proof-wait(33)	x							
-- --	held-for-diagnostics(34)	x							
-- --	service-off-line(35)	x							
-- --	no-space-on-server(36)	x							
--	pin-required(37)	x					x	x	x
-- --	exceeded-account-limit(38)	x					x	x	x
--	held-for-retry(39)	x							
--	job-printing(45)			x					

-- --		X/Open PSIS job-state-reasons extension values							
-- -- --	Descriptive Name	held	pending	processing	paused	interrupted	terminating	retained	completed
-- --	cancelled-by-shutdown(40)						x	x	x
-- --	device-unavailable(41)		x						
--	wrong-device(42)						x	x	x
--	bad-job(43)						x	x	x

-- --		X/Open PSIS job-state-reasons extension values							
-- -- --	Descriptive Name	held	pending	processing	paused	interrupted	terminating	retained	completed
-- --	job-interrupted-by-device-failure(44)	x							

-- textual-convention 4: JmAttributeTypeTC

```
JmAttributeTypeTC ::= TEXTUAL-CONVENTION
    STATUS      current
    DESCRIPTION
        "The type of the attribute.

        Attributes may represent information about a job, such as a
        file-name, or a document-name, or submission-time or completion
        time.  Attributes may also represent resources required, e.g., a
        medium or a colorant , etc. to process the job before the job
        start processing OR to indicate the amount of the resource that
        is being consumed while the job is processing, e.g., pages
        completed or impressions completed.  If both a required and a
        consumed value of a resource is needed, two separate attribute
        enums are assigned in the textual convention.

        In the following definitions of the enums, each description
        indicates whether the value of the attribute shall be
        represented using the jmAttributeValueAsInteger or the
        jmAttributeValueAsOctets objects by the initial tag: 'Integer:'
        or 'Octets:', respectively.  A very few attributes use both
        objects at the same time to represent a pair of values
        (mediumConsumed)and so have both tags.

        If the jmAttributeValueAsInteger object is not used (no
        'Integer:' tag), the agent shall return the value (-1)
        indicating other.  If the jmAttributeValueAsOctets object is not
        used (no "Octets:" tag), the agent shall return a zero-length
        octet string.

        The standard attribute types defined so far are:"

    -- This is a type 2 enumeration.  See Section 7.1 on page 29.
    SYNTAX      INTEGER {
     -- jm             Description - including Octets: or Integer:
     -- Attribute      to specify whether the value is represented
     -- TypeIndex      in the jmAttributeValueAsOctets or the
     --                jmAttributeValueAsInteger object,
     --                respectively.

     other(1),      -- An attribute that is not in the list and/or
                    -- that has not been registered with IANA.

     fileName(3),   -- Octets: The coded character set file name of
                    -- the document.
                    --
                    -- A row with this attribute item may appear
                    -- more than once in the jmAttributeTable for a
                    -- job.
```

```
documentName  -- Octets:  The coded character set name of the
(4),          -- document.
              --
              -- A row with this attribute item may appear
              -- more than once in the jmAttributeTable for a
              -- job.

jobAccountNa  -- Octets:  Arbitrary binary information which
me(5),        -- 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.

jobComment(6  -- 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 jmJobName object.

processingMe  -- Octets:  A coded character set message that
ssage(7),     -- is generated during the processing of the
              -- job as a simple form of processing log to
              -- show progress and any problems.
              --
              -- A row with this attribute item may appear
              -- more than once in the jmAttributeTable for a
              -- job.

jobSourceCha  -- Integer:  The index of the row in the
nnelIndex(8)  -- associated Printer MIB of the channel which
,             -- is the source of the print job.  See RFC
              -- 1759.
              --
              -- Must be 1 or greater.
              --
              -- NOTE - the Job Monitoring MIB points to the
              -- Channel row in the Printer MIB, so there is
              -- no need for a port object in the Job
              -- Monitoring MIB, since the PWG is adding a
              -- prtChannelInformation object to the Channel
              -- table of the draft Printer MIB.

outputBinInd  -- Integer:  The output subunit index in the
```

```
ex(9),          -- Printer MIB of the output bin to which all
                -- or part of the job is placed in.
                --
                -- A row with this attribute item may appear
                -- more than once in the jmAttributeTable for a
                -- job, but the jmAttributeValueAsInteger shall
                -- be different for each such row.

outputBinNam    -- Octets:  The name of the output bin to which
e(10),          -- all or part of the job is placed in.
                --
                -- A row with this attribute item may appear
                -- more than once in the jmAttributeTable for a
                -- job, but the jmAttributeValueAsOctets shall
                -- be different for each such row.

sides(11),      -- Integer:  The number of sides that any
                -- document in this job will require or did
                -- use.

documentForm    -- Integer:  The interpreter language family
atIndex(12),    -- index in the Printer MIB of the
                -- prtInterpreterLangFamily object, that this
                -- job requires and uses.  A document or a job
                -- may use more than one PDL.
                --
                -- A row with this attribute item may appear
                -- more than once in the jmAttributeTable for a
                -- job, but the jmAttributeValueAsInteger shall
                -- be different for each such row.  As with all
                -- intensive attribute items where multiple
                -- rows are allowed, there shall be only one
                -- distinct row for each distinct PDL; 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 as agent shall use the
                -- documentFormatEnum attribute instead.

documentForm    -- Integer:  The interpreter language family
atEnum(13),     -- corresponding to the Printer MIB
                -- prtInterpreterLangFamily object, that this
                -- job requires and uses.  A document or a job
                -- may use more than one PDL.
                --
                -- A row with this attribute item may appear
                -- more than once in the jmAttributeTable for a
                -- job, but the jmAttributeValueAsInteger shall
                -- be different for each such row.  As with all
                -- intensive attribute items where multiple
                -- rows are allowed, there shall be only one
                -- distinct row for each distinct PDL; there
                -- shall be no duplicates.
                --
                -- This enum is a type 2 enum.
                --
                -- NOTE: This textual convention is imported
                -- from the draft Printer MIB, but is not in
                -- RFC 1759.

physicalDevi    -- Integer:  The index of the physical device
ceIndex(14),    -- MIB instance requested/used, such as the
                -- Printer MIB.  This value is an hrDeviceIndex
                -- value.  See the Host Resource MIB.
                --
                -- A row with this attribute item may appear
                -- more than once in the jmAttributeTable for a
                -- job that is using more than one physical
                -- device, but the jmAttributeValueAsInteger
                -- shall be different for each such row.
                --
                -- If there is no physical device MIB instance
                -- for this job, this row shall not be present
                -- in the jmAttributeTable.

physicalDevi    -- Octets:  The name of the physical device to
ceName(15),     -- which the job is assigned.
                --
                -- A row with this attribute item may appear
                -- more than once in the jmAttributeTable for a
                -- job that is using more than one physical
                -- device, but the jmAttributeValueAsOctets
                -- shall be different for each such row.

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

jobCopiesReq    -- Integer:  The number of copies of the entire
uested(16),     -- job that are to be produce
                --
                -- A value of -2 means unknown.

jobCopiesCom    -- Integer:  The number of copies of the entire
pleted(17),     -- job that the entire job has completed so
                -- far.
                --
                -- A value of (-2) means unknown.

documentCopi    -- Integer:  The total count of the number of
esRequested(    -- document copies requested.  If there are
18),            -- 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.

documentCopi      -- Integer:  The total count of the number of
esCompleted(      -- document copies completed so far for the job
19),              -- 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.

jobKOctetsTo      -- Integer:  The total number of K (1024)
tal(20),          -- octets 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
                  -- preProcessing state and when the job is in
                  -- the held state with the jmJobStateReasons
                  -- object containing a documentsNeeded value,
                  -- the value of the jobKOctetsTotal 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).
                    --
                    -- The total K octets to be processed can be
                    -- used in the denominator with the
                    -- jmJobKOctetsCompleted attribute in the
                    -- numerator in order to produce a
                    -- 'thermometer' that indicates the progress of
                    -- the job.
                    --
                    -- The value (-2) means unknown.
                    --
    jobKOctetsCo    -- Integer:  The number of K (1024) octets
    mpleted(21),    -- currently processed by the device, including
                    -- document and job copies.  For printing, the
                    -- completed count includes processing
                    -- (interpreting) and marking.  For scanning,
                    -- the completed count include scanning.
                    --
                    -- 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
                    -- jobKOctetsTotal and the
                    -- jmJobKOctetsCompleted 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.
                    --
                    -- The total K octets completed can be used in
                    -- the numerator with the jobKOctetsTotal
                    -- attribute in the denominator in order to
                    -- produce a "thermometer" that indicates the
                    -- progress of the job.
                    --
                    -- The value of this attribute shall be 0 if
                    -- processing has not started for this 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.
-- ------------------------------------------------------
--
impressionsS  -- Integer:  The number of impressions spooled
pooled(22),   -- to the server or device for the job.

impressionsS  -- Integer:  The number of impressions sent to
entToDevice(  -- the device for the job.
23),
impressionsI  -- Integer:  The number of impressions
nterpreted(2  -- interpreted for the job.
4),
impressionsR  -- Integer:  The number of impressions
equested(25)  -- requested by this job to produce.
,
impressionsC  -- Integer:  The total number of impressions
ompleted(26)  -- completed by this job so far.
,             --
              -- The value of this attribute shall be 0 if
              -- processing has not started for this job.

impressionsC  -- Integer:  The number of impressions
ompletedCurr  -- completed for the current copy of the
entCopy(27),  -- current document.
              --
              -- The value of this attribute shall be 0 if
              -- processing has not started for this job.

-- ------------------------------------------------------------
-- 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.
-- ------------------------------------------------------------
--
pagesRequest  -- Integer:  The number of logical pages
ed(28),       -- requested by the job to be processed.

pagesComplet  -- Integer:  The total number of logical pages
ed(29),       -- completed for this job.

pagesComplet  -- Integer:  The number of logical pages
edCurrentCop  -- completed for the current copy of the
y(30),        -- document.  This value is reset to 0 for each
              -- document and for each document copy.

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

sheetsReques  -- Integer:  The total number of medium sheets
ted(31),      -- requested to be processed for this job.
```

```
sheetsComple  -- Integer:  The total number of medium sheets
ted(32),      -- that have been completed for the entire job
              -- whether those sheets have been processed on
              -- one side or on both.
              -- The value of this attribute shall be 0 if
              -- processing has not started for this job.

sheetsComple  -- Integer:  The number of medium sheets that
tedCurrentCo  -- have been completed for the current copy of
py(33),       -- a document in the job whether those sheets
              -- have been processed on one side or on both.
              -- The value of this attribute shall be 0 if
              -- processing has not started for this job.

mediumReques  -- Octets:  The name of the medium that is
ted(34),      -- required by the job.
              --
              -- A row with this attribute item may appear
              -- more than once in the jmAttributeTable for a
              -- job, but the jmAttributeValueAsOctets shall
              -- be different for each such row.

mediumConsum  -- Octets: The name of the medium AND
ed(35),       --
              -- Integer:  the number of sheets that have
              -- been consumed whether those sheets have been
              -- processed on one side or on both.  This
              -- attribute shall have both values.
              --
              -- A row with this attribute item may appear
              -- more than once in the jmAttributeTable for a
              -- job, but the jmAttributeValueAsOctets shall
              -- contain a different name for each such row.
              --
              -- The value of this attribute shall be 0 if
              -- processing has not started for this job.
              --
colorantRequ  -- Integer:  The index (prtMarkerColorantIndex)
estedIndex(3  -- in the Printer MIB of the colorant
6),           -- requested.
              --
              -- A row with this attribute item may appear
              -- more than once in the jmAttributeTable for a
              -- job, but the jmAttributeValueAsOctets shall
              -- be different for each such row.

colorantRequ  -- Octets:  The name of the colorant requested.
estedName(37  --
),            -- A row with this attribute item may appear
              -- more than once in the jmAttributeTable for a
              -- job, but the jmAttributeValueAsOctets shall
              -- be different for each such row.
```

```
               --
colorantCons   -- Integer:  The index (prtMarkerColorantIndex)
umedIndex(38   -- in the Printer MIB of the colorant consumed.
),             --
               -- A row with this attribute item may appear
               -- more than once in the jmAttributeTable for a
               -- job, but the jmAttributeValueAsOctets shall
               -- be different for each such row.

colorantCons   -- Octets:  The name of the colorant consumed.
umedName(39)   --
,              -- A row with this attribute item may appear
               -- more than once in the jmAttributeTable for a
               -- job, but the jmAttributeValueAsOctets shall
               -- be different for each such row.
               --
-- -------------------------------------------------------------
-- Time attributes:  two forms of time are provided:
-- DateAndTime and TimeStamp from SNMPv2TC (RFC 1903).
-- DateAndTime is an 8 or 11 octet binary encoded year,
-- month, day, hour, minute, second, deci-second with
-- optional offset from UTC.  TimeStamp is the integer value
-- of sysUpTime (in hundredths of a second).  See page 32.
-- -------------------------------------------------------------

jobSubmissio   -- Octets:  The date and time that the job was
nDateAndTime   -- submitted.  The value shall be specified
(40),          -- using the DateAndTime textual convention
               -- from SMIv2-TC (see page 32).
               --
               -- NOTE: DateAndTime is not printable
               -- characters.

jobSubmissio   -- Integer:  The time that the job was
nTimeStamp(4   -- submitted.  The value shall be specified
1),            -- using the TimeStamp textual convention from
               -- SMIv2-TC (see page 32).

jobStartedPr   -- Octets:  The date and time that the job
ocessingDate   -- started processing.  The value shall be
AndTime(42),   -- specified using the DateAndTime textual
               -- convention from SMIv2-TC (see page 32).

jobStartedPr   -- Integer:  The time that the job started
ocessingTime   -- processing.  The value shall be specified
Stamp(43),     -- using the TimeStamp textual convention from
               -- SMIv2-TC (see page 32).

jobCompleted   -- Octets:  The date and time that the job
DateAndTime(   -- completed processing and the medium is
44),           -- completely stacked in the output bin.  The
               -- value shall be specified using the
               -- DateAndTime textual convention from SMIv2-TC
```

```
                    -- (see page 32).

    jobCompleted    -- Integer:  The time that the job completed
    TimeStamp(45    -- processing and the medium is completely
    ),              -- stacked in the output bin.  The value shall
                    -- be specified using the TimeStamp textual
                    -- convention from SMIv2-TC (see page 32).
                    --
    processingCP    -- Integer:  The amount of CPU time that the
    UTime(46)       -- job has been processing in seconds.  If the
                    -- job needs attention, that elapsed time shall
                    -- not be included.  In other words, the
                    -- processingCPUTime should be relatively
                    -- repeatable.
                    --
                    -- The value of this attribute shall be 0 if
                    -- processing has not started for this job.

996     }
```

```
--  The General Group (Mandatory)
--
--  The jmGeneralGroup consists of information of a general nature
--  that are per-job-set, but are not per-job.  The jmGeneralGroup
--  consists entirely of the jmGeneralEntry which is indexed by:
--
--  1) jmJobSetIndex -  a running index of Job Set
--     instances supported by this device or server.  A
--     job set is used in the MIB to represent the
--     separation of jobs into disjoint sets for
--     scheduling purposes in a server, typically into
--     separate job queues.  See Terminology and Job Model
--     on page 11 for the definition of a job set.
--
--  Implementation of every object in this group is mandatory.  See
--  Section 4 entitled 'Conformance Considerations' on page 27.
--

jmGeneral  OBJECT IDENTIFIER ::= { jobmonmib 5 }

jmGeneralTable  OBJECT-TYPE
    SYNTAX      SEQUENCE OF JmGeneralEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A table of general information per-job-set ( queue), but not
        per-job.  See Terminology and Job Model on page 11 for the
        definition of a job set."
    ::= { jmGeneral 1 }

jmGeneralEntry  OBJECT-TYPE
    SYNTAX      JmGeneralEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "Information about a job set (queue).  See Terminology and Job
        Model on page 11 for the definition of a job set.

        An entry shall exist in this table for each job set."
    INDEX  { jmJobSetIndex }
    ::= { jmGeneralTable 1 }

JmGeneralEntry ::= SEQUENCE {
    jmJobSetIndex                     Integer32(1..32767),
    jmGeneralJobSetName               OCTET STRING(SIZE(0..63))
    jmGeneralJobCompletedPolicy       Integer32(0..2147483647),
    jmGeneralMaxNumberOfJobs          Integer32(0..2147483647),
    jmGeneralNumberOfJobsToComplete   Integer32(0..2147483647),
    jmGeneralNumberOfJobsCompleted    Integer32(0..2147483647)
}

jmJobSetIndex OBJECT-TYPE
```

```
    SYNTAX      Integer32(1..32767)
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The 16-bit index of a Job Set instance used to represent the
        separation of jobs into disjoint sets for scheduling purposes in
        a server, typically into separate job queues.  See Terminology
        and Job Model on page 11 for the definition of a job set.
        Agents implementing a single Job Set instance shall use an index
        value of 1 for this object."
    ::= { jmGeneralEntry 1 }

jmGeneralJobSetName OBJECT-TYPE
    SYNTAX      OCTET STRING(SIZE(0..63))
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The human readable administratively assigned name of this job
        set.  Typically, this name will be the name of the job queue.
        If a server or printer has only a single job set, this object
        can be the administratively assigned name of the server or
        printer itself.  This name does not need to be unique, though
        each job set in a single Job Monitoring MIB should have distinct
        names.

        The purpose of this object is to help the user of the job
        monitoring application distinguish between several job sets in
        implementations that support more than one job set."
    ::= { jmGeneralEntry 2 }

jmGeneralJobCompletedPolicy OBJECT-TYPE
    SYNTAX      Integer32(0..2147483647)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The time in seconds that the device or server keeps jobs in the
        jmJobTable and jmJobCompletedTable after processing as specified
        by the system administrator for this instance of the Job Set."
    ::= { jmGeneralEntry 3 }

jmGeneralMaxNumberOfJobs OBJECT-TYPE
    SYNTAX      Integer32(0..2147483647)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The maximum number of queued and completed jobs that this
        server or print can support at the same time.

        The value (-1) indicating other shall indicate that there is no
        fixed limit."
    ::= { jmGeneralEntry 4 }

jmGeneralNumberOfJobsToComplete OBJECT-TYPE
```

```
    SYNTAX      Integer32(0..2147483647)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The total number of jobs currently in the jmJobTable that are
        to be completed, i.e., the total number of jobs that are in the
        following states: pre-processing, held, pending, processing,
        needs-attention, paused, interrupted, or terminating, but not
        retained or completed.  See JmJobStateTC on page 38 for the
        exact specification of the semantics of the job states."
    ::= { jmGeneralEntry 5 }

jmGeneralNumberOfJobsCompleted OBJECT-TYPE
    SYNTAX      Integer32(0..2147483647)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The total number of jobs currently in the jmJobTable that are
        completed, i.e., the total number of jobs that are in the
        following states: retained or completed, but not pre-processing,
        held, pending, processing, needs-attention, paused, interrupted,
        or terminating.  See JmJobStateTC on page 38 for the exact
        specification of the semantics of retained, completed and the
        other states.

        The value of the jmGeneralNumberOfJobsCompleted shall equal the
        number of jobs in the jmCompletedTable.  The sum of
        jmGeneralNumberOfJobsToComplete and
        jmGeneralNumberOfJobsCompleted shall be equal to the number of
        jobs in the jmJobTable."
    ::= { jmGeneralEntry 6 }
```

```
-- The Queue Group (Conditionally Mandatory)
--
-- The jmQueueGroup consists of job objects that are needed by a
-- server or device that queues jobs, but are not needed after the
-- job has completed processing, i.e., are not needed by accounting
-- applications.
--
-- The jmQueueGroup is conditionally mandatory meaning that the
-- jmQueueGroup shall be implemented by a Job Monitoring MIB agent
-- that is instrumenting a server or printer that performs queuing
-- (or spooling).
--
-- The jmQueueGroup is made up entirely of the jmQueueTable which is
-- an ordered list of jobs in a job set that have not completed
-- processing.  The jmQueueTable is indexed by:
--
-- 1) jmJobSetIndex -  a running index of Job Set instances supported
--    by this device or server.  A job set is used in the MIB to
--    represent the separation of jobs into disjoint sets for
--    scheduling purposes in a server, typically into separate job
--    queues.  See Terminology and Job Model on page 11 for the
--    definition of a job set.
--
-- 2) jmQueueIndex - a running index of the jobs that have not
--    finished processing and shall indicate the order that the jobs
--    are currently scheduled to be processed.
--
-- Implementation of this group is conditionally mandatory, i.e.,
-- mandatory if the server or printer that the agent is instrumenting
-- queues jobs (rather than just passing the jobs through).  See
-- Section 4 entitled 'Conformance Considerations' on page 27.

jmQueue  OBJECT IDENTIFIER ::= { jobmonmib 6 }

jmQueueTable  OBJECT-TYPE
    SYNTAX      SEQUENCE OF JmQueueEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A table of per-job information needed by a server or device
        that performs queuing."
    ::= { jmQueue 1 }

jmQueueEntry  OBJECT-TYPE
    SYNTAX      JmQueueEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "Information about a job in a server or printer that performs
        queuing.

```

```
            An entry shall exist in this table for each job in a job set
            that is queued, i.e., for each job that has not completed
            processing."
        INDEX  { jmJobSetIndex, jmQueueIndex }
        ::= { jmQueueTable 1 }

    JmQueueEntry ::= SEQUENCE {
        jmQueueIndex                      Integer32(1..2147483647),
        jmQueueJobIndex                   Integer32(1..2147483647),
        jmQueueNumberOfInterveningJobs    Integer32(0..2147483647),
        jmJobPriority                     Integer32(0..100),
        jmJobProcessAfterDateAndTime      DateAndTime
    }

    jmQueueIndex OBJECT-TYPE
        SYNTAX      Integer32(0..2147483647)
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
            "The 32-bit index of the jobs that have not finished processing.
            The index values shall be assigned monatonically increasing as
            the server or printer determines the order of processing.  The
            agent shall change the value of this object dynamically as the
            priority ordering of jobs changes.  Thus the jmQueueTable orders
            the jobs into their current priority order which can change as
            new jobs are submitted and/or the configuration of the Printer
            is changed."
        ::= { jmQueueEntry 1 }

    jmQueueJobIndex OBJECT-TYPE
        SYNTAX      Integer32(1..2147483647)
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
            "The job's identifier generated by the server or device when
            that server or device accepted the job.  This value permits the
            management application to access the other tables to obtain the
            job-specific objects.  This value shall be the same for a job in
            the jmQueueTable as the corresponding jmJobIndex value in the
            jmJobTable for this job.

            The value 0 shall not be generated.  Agents instrumenting
            systems that contain jobs with a job identifier of 0 shall map
            the value 0 to a value that is one higher than the highest job
            identifier value that any job can have on that system."
        ::= { jmQueueEntry 2 }

    jmQueueNumberOfInterveningJobs OBJECT-TYPE
        SYNTAX      Integer32(0..2147483647)
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "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.  The agent
            shall return a value of 0 for this object when the job starts
            processing."
        ::= { jmQueueEntry 3 }

    jmJobPriority OBJECT-TYPE
        SYNTAX      Integer32(0..100)
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "This attribute specifies a 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.

            A value of 0 shall be returned by implementations that do not
            have a priority-based queuing algorithm."
        ::= { jmQueueEntry 4 }

    jmJobProcessAfterDateAndTime OBJECT-TYPE
        SYNTAX      DateAndTime
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "This object specifies 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 jmJobCurrentState to
            held and add the jobProcessAfterSpecified bit value to the job's
            jmJobStateReasons object 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
            jmJobStateReasons object and, if no other reasons remain, shall
            change the job's jmJobCurrentState to pending so that the job
            becomes a candidate for being scheduled on devices(s).

            The server shall assign an empty value to the
            jmJobProcessAfterDateAndTime object when no process after time
            has been specified, so that the job shall be a candidate for
            processing immediately."
        ::= { jmQueueEntry 5 }
```

```
-- The Completed Group (Mandatory)
--
-- The jmCompletedGroup consists entirely of the jmCompletedTable
-- which is an ordered list of the jobs in the job set that have
-- completed processing, i.e., jobs that are in the terminating,
-- retained or completed state.  The jmCompletedTable is indexed by:
--
-- 1) jmJobSetIndex -  a running index of Job Set instances supported
--    by this device or server.  A job set is used in the MIB to
--    represent the separation of jobs into disjoint sets for
--    scheduling purposes in a server, typically into separate job
--    queues.  See Terminology and Job Model on page 11 for the
--    definition of a job set.
--
-- 2) jmCompletedIndex - a running index of the jobs that have
--    finished processing.
--
-- Implementation of every object in this group is mandatory.  See
-- Section 4 entitled 'Conformance Considerations' on page 27.

jmCompleted  OBJECT IDENTIFIER ::= { jobmonmib 7 }

jmCompletedTable  OBJECT-TYPE
    SYNTAX      SEQUENCE OF JmCompletedEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A table of pointers to jobs that have finished processing, have
        been cancelled by a user or operator, or the system has
        aborted."
    ::= { jmCompleted 1 }

jmCompletedEntry  OBJECT-TYPE
    SYNTAX      JmCompletedEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A pointer to a job that has finished processing.

        An entry shall exist in this table for each job that has
        finished processing, due to normal completion, cancellation by a
        user, or termination by the system."
    INDEX  { jmJobSetIndex, jmCompletedIndex }
    ::= { jmCompletedTable 1 }

JmCompletedEntry ::= SEQUENCE {
jmCompletedIndex              Integer32(1..2147483647),
jmCompletedJobIndex           Integer32(1..2147483647)
}

jmCompletedIndex OBJECT-TYPE
    SYNTAX      Integer32(1..2147483647)
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The 32-bit index of the jobs that are in the retained or
        completed states.  The agent shall add jobs to the end of the
        jmCompletedTable, so that monitor programs can quickly determine
        what jobs have completed since the last time that the monitoring
        programs accessed the jmCompletedTable.  The index values shall
        be monatonically increasing.  Therefore, the order of the jobs
        specified by the value of this index shall be the order in which
        the jobs finished processing.

        Since the jmCompletedIndex shall roll over when the
        jmCompletedIndex would have reached 2^31 (but no lower),
        monitoring programs shall handle such roll over."
    ::= { jmCompletedEntry 1 }

jmCompletedJobIndex OBJECT-TYPE
    SYNTAX      Integer32(1..2147483647)
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The job's identifier generated by the server or device when
        that server or device accepted the job.  This value permits the
        management application to access the other tables to obtain the
        job-specific objects.  This value shall be the same for a job in
        the jmQueueTable as the corresponding jmJobIndex value in the
        jmJobTable for this job.

        The value 0 shall not be generated.  Agents instrumenting
        systems that contain jobs with a job identifier of 0 shall map
        the value 0 to a value that is one higher than the highest job
        identifier value that any job can have on that system."
    ::= { jmCompletedEntry 2 }
```

```
-- The Job Group (Mandatory)
--
-- The jmJobGroup consists of basic job identification and status
-- information for each job in a job set that (1) monitoring
-- applications need to be able to access in a single SNMP Get
-- operation, (2) that have a single value per job, and (3) that
-- shall always be implemented.
--
-- The jmJobGroup consists entirely of the jmJobTable which is
-- indexed by:
--
-- 1) jmJobSetIndex -  a running index of Job Set instances supported
--    by this device or server.  A job set is used in the MIB to
--    represent the separation of jobs into disjoint sets for
--    scheduling purposes in a server, typically into separate job
--    queues.  See Terminology and Job Model on page 11 for the
--    definition of a job set.
--
-- 2) jmJobIndex -  the job identifier that was generated by the
--    server or device that accepted the job.
--
-- Implementation of every object in this group is mandatory.  See
-- Section 4 entitled 'Conformance Considerations' on page 27.

jmJob  OBJECT IDENTIFIER ::= { jobmonmib 8 }

jmJobTable  OBJECT-TYPE
    SYNTAX      SEQUENCE OF JmJobEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A table of basic job identification and status information for
        each job in a job set."
    ::= { jmJob 1 }

jmJobEntry  OBJECT-TYPE
    SYNTAX      JmJobEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "Basic per-job identification and status information.

        An entry shall exist in this table for each job, no matter what
        the state of the job is.  Each job shall appear in one and only
        one job set."
    INDEX  { jmJobSetIndex, jmJobIndex }
    ::= { jmJobTable 1 }

JmJobEntry ::= SEQUENCE {
-- Job Identification (I) objects:
    jmJobIndex                        Integer32(1..2147483647),
    jmJobName                         OCTET STRING(SIZE(0..63)),
    jmJobIdName                       OCTET STRING(SIZE(0..63)),
    jmJobIdNumber                     Integer32(0..2147483647),
    jmJobServiceTypes                 Integer32(1..2147483647),
                                      -- JmJobServiceTypesTC
    jmJobOwner                        OCTET STRING(SIZE(0..63)),
    jmJobDeviceNameOrQueueRequested   OCTET STRING(SIZE(0..63)),

-- Job Status (S) objects:
    jmJobCurrentState                 JmJobStateTC,
    jmJobStateReasons                 OCTET STRING(SIZE(0..63))
                                      -- encoded as a bit string
}


-- Job Identification (I) objects
--
-- The following jmJobGroup objects identify the job to the user of
-- the management application which may be acting in the role of an
-- end-user or a system operator:

jmJobIndex OBJECT-TYPE
    SYNTAX      Integer32(1..2147483647)
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The identifier of the job on the device or server.   The job's
        identifier is generated by the server or device when that server
        or device accepted the job.  However, if the device does not
        generate a job identifier for each job, then the Job Monitoring
        MIB agent shall generate the job identifier for the job.

        The value 0 shall not be generated.  Agents instrumenting
        systems that contain jobs with a job identifier of 0 shall map
        the value 0 to a value that is one higher than the highest job
        identifier value that any job can have on that system."
    ::= { jmJobEntry 1 }

jmJobName OBJECT-TYPE
    SYNTAX      OCTET STRING(SIZE(0..63))
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "This object is 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.

        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(4) resource item
        of the first document in the job or the fileName(3) resource
        item of the first document in the job.

        The jmJobName is distinguished from the jobComment attribute, in
        that the jmJobName 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."
    ::= { jmJobEntry 2 }

jmJobIdName OBJECT-TYPE
    SYNTAX      OCTET STRING(SIZE(0..63))
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "Identifies the job on the "client-side" of the printing process
        as coded character set data in combination with the
        jmJobIdNumber object.

        The jmJobIdName and the jmJobIdNumber objects are referred to as
        the "client-side" identifiers because they allow the user,
        operator, or the system administrator to uniquely identify the
        print jobs of interest from all the jobs currently "known" by
        the server or device.

        The client-side identifiers can be assigned by either the job
        submission client's local system or a downstream server,
        depending on implementation and the job submission protocol.
        The format of the coded character set data and point of
        assignment of the client-side identifiers depend upon the job
        submission protocol in use.  See Appendix A on page 87 for the
        mapping from selected job submission protocols to these client-
        side job identifiers.

        Unlike jmJobName, which is assigned by the submitting user, the
        jmJobIdName and jmJobIdNumber client-side identifiers provide
        for unique identification of jobs.

        The jmJobIdName object may be used alone or in conjunction with
        the jmJobIdNumber object, depending upon the format of the job
        submission protocol client side identifier.  For example, the
        LPD job identifier normally contains three alpha characters
        followed by a three digit number.  The agent may represent the
        alpha portion by jmJobIdName and the numeric portion by
        jmJobIdNumber.  Alternatively, the agent may represent the LPD
        client-side id entirely in the jmJobIdName object."
    ::= { jmJobEntry 3 }
```

```
jmJobIdNumber OBJECT-TYPE
    SYNTAX      Integer32(0..2147483647)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "Identifies the job on the "client-side" of the printing process
        in combination with the jmJobIdName object. This object may be
        used alone or in conjunction with the jmJobIdName  object,
        depending upon the format of the job submission protocol client-
        side identifier.  Refer to the jmJobIdName object specification.

        If the value of this object is unknown, the agent shall return
        the value (-2)."
    ::= { jmJobEntry 4 }

jmJobServiceTypes OBJECT-TYPE
    SYNTAX      Integer32(1..2147483647)   --See JmJobServiceTypesTC on
    page 36
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "Specifies the type(s) of service to which the job has been
        submitted (print, fax, scan, etc.).  The service type is
        represented as an enum that 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, fax, and print a single job.  In this case,
        three bits would be set in the jmJobServiceTypes object,
        corresponding to the values: 8+32+4=44, respectively.

        Whether this object 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.  With
        either implementation, the agent shall return a non-zero value
        for this object indicating the type of the job.

        One of the purposes of this object 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 the object is in the job identification
        category.

        This object is a type 2 enum.

        The JmJobServiceTypesTC textual convention defines component
        types as separate bit value in the enum.  See page 36."
    ::= { jmJobEntry 5 }

jmJobOwner OBJECT-TYPE
    SYNTAX      OCTET STRING(SIZE(0..63))
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "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."
    ::= { jmJobEntry 6 }

jmJobDeviceNameOrQueueRequested OBJECT-TYPE
    SYNTAX      OCTET STRING(SIZE(0..63))
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The administratively defined coded character set name of the
        target device or queue.  Its value corresponds to the Printer
        MIB: prtGeneralAdminName object (added to the draft Printer MIB)
        for printers.  For servers, this object is the name that users
        supply to indicate whether they want the job to be processed,
        typically, but not limited to, a job queue name or logical
        printer name."
    ::= { jmJobEntry 7 }

jmJobCurrentState OBJECT-TYPE
    SYNTAX      JmJobStateTC   -- See page 38
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The current state of the job (pending, processing, held, etc.)

        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.

        A companion textual convention (JmJobStateReasonsTC) and
        corresponding object (jmJobStateReasons) provide additional
        information about job states.  While the job states cannot be
        added to without impacting deployed clients, it is the intent
        that additional JmJobStateReasonsTC enums can be defined without
        impacting deployed clients.  In other words, the
        JmJobStateReasonsTC is intended to be extensible.  See page 42.

        This object is a type 2 enum."
    ::= { jmJobEntry 8 }

jmJobStateReasons OBJECT-TYPE
    SYNTAX      OCTET STRING(SIZE(0..63))  -- encoded as a bit string
                                           -- See JmJobStateReasonsTC
                                           -- on page 42
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "This object provides additional information regarding the
        jmJobCurrentState object.  This object identifies the reason or
        reasons that the job is in the preProcessing, held, pending,
        processing, needsAttention, paused, interrupted, terminating,
        retained, or completed state.  The server shall indicate the
        particular reason(s) by setting the value of the
        jmJobStateReasons object.  While the job states cannot be added
        to without impacting deployed clients, it is the intent that
        additional JmJobStateReasonsTC enums can be defined without
        impacting deployed clients.  In other words, the
        JmJobStateReasonsTC is intended to be extensible.  See page 42.

        When the job does not have any reasons for being in its current
        state, the server shall set the value of the jmJobStateReasons
        object  to a bit string containing all zeros.

        Bits in the bit string are assigned starting with the most
        significant bit in the most significant octet which is called
        bit 1.  Bit 2 is the next most significant bit in the most
        significant octet, etc.  Bit 9 is the most significant bit in
        the second most significant octet, etc., up to the maximum bit:
        504 (= 8 x 63).  See JmJobStateReasonsTC on page 42

        An agent only need return the most significant octet up to the
        least significant octet that contains a non-zero bit.

        If all bits are zero, the agent may return an OCTET STRING of
        zero length.  Alternatively, an agent may always return a fixed
        number of octets starting with the most significant octet and
        running through the least significant octet that could ever have
        a one bit in it for that implementation.

        This object is a type 2 bit string.  See Section 7 entitled
        'IANA Considerations' on page 29 and Section 12 entitled
        'Datatypes used in the Job Monitoring MIB' on page 32."
    ::= { jmJobEntry 9 }
```

```
-- The Attribute Group (Mandatory)
--
-- The jmAttributeGroup consists attributes of the job and
-- document(s).  Attribute may represent information about the job
-- and document(s), such as file-names, document-names, submission-
-- time, completion-time, size.  Attributes may also represent
-- requested and/or consumed resources for each job.  Instead of
-- allocating distinct objects for each attribute, each attribute
-- item is represented as a separate row in the jmAttributeTable.
-- Each column in the row describes the attribute, such as its type
-- represented as an enum, and the value represented as (1) an
-- integer or (2) an octet string (character coded text and binary
-- octet strings, such as DateAndTime) or (3) both.
--
-- Most attribute items shall have only one row per job.  However, a
-- few attribute items can have multiple values per job or even per
-- document, where each value is a separate row in the
-- jmAttributeTable.  Unless indicated otherwise, an agent shall
-- ensure that each attribute item occurs only once in the
-- jmAttributeTable.  Attribute items that may appear multiple times
-- in the jmAttributeTable are indicated in their specification in
-- the JmAttributeTypeTC (see page 54).  However, such attribute
-- items shall not contain duplicates for "intensive" (as opposed to
-- "extensive") attributes.  For example, each documentFormat(11)
-- shall appear in the jmAttributeTable only once for a job since the
-- interpreter language is an intensive attribute item, even though
-- the job has a number of documents that all use the same PDL.  As
-- another example of an intensive attribute that can have multiple
-- entries, if a document or job uses multiple types of media, there
-- shall be only one row in the jmAttributeTable for each media type,
-- not one row for each document that uses that medium type.  On the
-- other hand, if a job contains two documents of the same name,
-- there can be separate rows for the documentName(4) attribute item
-- with the same name, since a document name is an extensive
-- attribute item.
--
-- The jmAttributeGroup consists entirely of the jmAttributeTable
-- which is indexed by (from most significant to least significant):
--
-- 1) jmJobSetIndex -  a running index of Job Set instances supported
--    by this device or server.  A job set is used in the MIB to
--    represent the separation of jobs into disjoint sets for
--    scheduling purposes in a server, typically into separate job
--    queues.  See Terminology and Job Model on page 11 for the
--    definition of a job set.
--
-- 2) jmJobIndex -  the job identifier that was generated by the
--    server or device that accepted the job.
--
-- 3) jmAttributeTypeIndex - the enum that indicates the type of
--    attribute.  See JmAttributeTypeTC on page 54.
--
```

```
-- 4) jmAttributeInstanceIndex -  a running index of attributes of the
--    same type for each job.  For those attributes with only a single
--    instance per job, this index value shall be 1.  For those
--    attributes that are a single value per document, the index value
--    shall be the document number, starting with 1 for the first
--    document in the job.  Jobs with only a single document shall use
--    the index value of 1.  For those attributes that can have
--    multiple values per job and per document, such as
--    documentFormatIndex or documentFormatEnum, the index shall be a
--    running index for the job as a whole, starting at 1.
--
-- The jmAttributeTable is a per job table with an extra index for
-- each type of attribute (jmAttributeTypeIndex) that a job can have
-- and an additional index (jmAttributeInstanceIndex)for those
-- attributes that can have multiple instances per job.  The
-- jmAttributeTypeIndex object shall contain an enum type that
-- indicates the type of attribute.  Some attribute types are used to
-- represent a resources that is both requested and consumed as a
-- single value, depending on the point in time, while other
-- attributes have distinct types for requested versus consumed
-- values.  The agent is able to discover the attributes either from
-- the job submission protocol itself or from the document PDL.  As
-- the documents are interpreted, the interpreter may discover
-- additional attributes and so adds additional rows to this table.
-- As the resources are actually consumed, the usage counter
-- contained in the jmAttributeValueAsInteger object is incremented
-- according to the units indicated in the description of the enum.
-- See JmAttributeTypeTC on page 54.
--
-- Some attributes are mandatory for conformance, and the rest are
-- optional.  The mandatory attributes are:
--
--     sheetsCompleted(14)
--
-- Implementation of every object in this group is mandatory.  See
-- Section 4 entitled 'Conformance Considerations' on page 27.

jmAttribute  OBJECT IDENTIFIER ::= { jobmonmib 9 }

jmAttributeTable  OBJECT-TYPE
    SYNTAX      SEQUENCE OF JmAttributeEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A table of attributes for each job in a job set.  Attributes
        may represent information about the job and document(s) or
        resources required and/or consumed."
    ::= { jmAttribute 1 }

jmAttributeEntry  OBJECT-TYPE
    SYNTAX      JmAttributeEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "Attributes representing information about the job and
        document(s) or resources required and/or consumed.

        Zero or more entries shall exist in this table for each job in a
        job set.  Each job shall appear in one and only one job set."
    INDEX  { jmJobSetIndex, jmJobIndex, jmAttributeTypeIndex,
    jmAttributeInstanceIndex }
    ::= { jmAttributeTable 1 }

JmAttributeEntry ::= SEQUENCE {
    jmAttributeTypeIndex          JmAttributeTypeTC,
    jmAttributeInstanceIndex      Integer32(1..32767),
    jmAttributeValueAsInteger     Integer32(0..2147483647),
    jmAttributeValueAsOctets      OCTET STRING(SIZE(0..63))
}

jmAttributeTypeIndex OBJECT-TYPE
    SYNTAX      JmAttributeTypeTC   -- See page 54
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The type of attribute.

        The type may identify information about the job or document(s)
        or may identify a resource required to process the job before
        the job start processing and/or consumed by the job as the job
        is processed.

        Examples of job and document information include:
        jobCopiesRequested, documentCopiesRequested, jobCopiesCompleted,
        documentCopiesCompleted, fileName, and documentName.

        Examples of resources required and consumed include:
        jobKOctetsTotal, jobKOctetsCompleted, pagesRequested,
        pagesCompleted, mediumRequested, and mediumConsumed.  See the
        JmAttributeTypeTC textual convention on page 54.

        In the definitions of the enums in the JmAttributeTypeTC textual
        convention, each description indicates whether the value of the
        attribute shall be represented using the
        jmAttributeValueAsInteger or the jmAttributeValueAsOctets
        objects by the initial tag: "Integer:" or "Octets:",
        respectively.  A very few attributes use both objects
        (mediumConsumed)and so have both tags.

        If the jmAttributeValueAsInteger object is not used (no
        "Integer:" tag), the agent shall return the value (-1)
        indicating other.  If the jmAttributeValueAsOctets object is not
        used (no "Octets:" tag), the agent shall return a zero-length
        octet string.

        This value is a type 2 enum."
    ::= { jmAttributeEntry 1 }

jmAttributeInstanceIndex OBJECT-TYPE
    SYNTAX      Integer32(1..32767)
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A running 16-bit index of the attributes of the same type for
        each job.  For those attributes with only a single instance per
        job, this index value shall be 1.  For those attributes that are
        a single value per document, the index value shall be the
        document number, starting with 1 for the first document in the
        job.  Jobs with only a single document shall use the index value
        of 1.  For those attributes that can have multiple values per
        job and per document, such as documentFormatIndex or
        documentFormatEnum, the index shall be a running index for the
        job as a whole, starting at 1.

        Each job shall be identified by jmJobIndex value and each job
        shall be in one job set identified by jmJobSetIndex."
    ::= { jmAttributeEntry 2 }

jmAttributeValueAsInteger OBJECT-TYPE
    SYNTAX      Integer32(0..2147483647)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The integer value of the attribute.  The value of the attribute
        shall be represented as an integer if the enum description
        JmAttributeTypeTC definition (see JmAttributeTypeTC on page 54)
        has the tag: 'Integer:'.

        Depending on the enum definition, this object value may be an
        integer, a counter, an index, or an enum, depending on the
        jmAttributeTypeIndex value.  The units of this value are
        specified in the enum description.

        For those attributes that are accumulating job consumption as
        the job is processed as specified in the JmAttributeTypeTC,
        shall contain the final value after the job completes
        processing, i.e., this value shall indicate the total usage of
        this resource made by the job.

        A monitoring application is able to copy this value to a
        suitable longer term storage for later processing as part of an
        accounting system.

        Since the agent may add attributes representing resources to
        this table while the job is waiting to be processed or being
        processed, which can be a long time before any of the resources
        are actually used, the agent shall set the value of the
        jmAttributeValueAsInteger object to 0 for resources that the job
        has not yet consumed.
        Attributes for which the concept of an integer value is
        meaningless, such as fileName, interpreter, and physicalDevice,
        do not have the 'Integer:' tag in the JmAttributeTypeTC
        definition and so shall return a value of (-1) to indicate other
        for jmAttributeValueAsInteger."
    ::= { jmAttributeEntry 3 }

jmAttributeValueAsOctets OBJECT-TYPE
    SYNTAX      OCTET STRING(SIZE(0..63))
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The octet string value of the attribute.  The value of the
        attribute shall be represented as an OCTET STRING if the enum
        description JmAttributeTypeTC definition (see JmAttributeTypeTC
        on page 54) has the tag: 'Octets:'.

        Depending on the enum definition, this object value may be a
        coded character set string (text) or a binary octet string, such
        as DateAndTime.

        Attributes for which the concept of an octet string value is
        meaningless, such as pagesCompleted, do not have the tag
        'Octets:' in the JmAttributeTypeTC definition and so shall
        return a value of a zero length string for
        jmAttributeValueAsOctets."
    ::= { jmAttributeEntry 4 }
```

```
-- Conformance Information

jmMIBConformance OBJECT IDENTIFIER ::= { jobmonmib 2 }

-- compliance statements
jmMIBCompliance MODULE-COMPLIANCE
    STATUS  current
    DESCRIPTION
        "The compliance statement for agents that implement the
        job monitoring MIB."
    MODULE -- this module
    MANDATORY-GROUPS {
        jmGeneralGroup, jmCompletedGroup, jmJobGroup, jmAttributeGroup }

        OBJECT  jmJobCurrentState
        SYNTAX      INTEGER {
              processing(7),
              needsAttention(9),
              completed(17)
        }
    DESCRIPTION
        "It is conformant for an agent to implement just these three
        states in this object.  Any additional states are optional.
        However, a client shall accept all of the states from an agent."

    --  the jmQueueGroup is conditionally mandatory.  An agent shall
    --  implement the jmQueueGroup if the server or device that the
    --  agent instruments performs queuing.
    ::= { jmMIBConformance 1 }

jmMIBGroups      OBJECT IDENTIFIER ::= { jmMIBConformance 2 }

jmGeneralGroup OBJECT-GROUP
    OBJECTS {
        jmGeneralJobSetName, jmGeneralJobCompletedPolicy,
        jmGeneralMaxNumberOfJobs, jmGeneralNumberOfJobsToComplete,
        jmGeneralNumberOfJobsCompleted,  }
    STATUS  current
    DESCRIPTION
        "The general group."
    ::= { jmMIBGroups 1 }

jmQueueGroup OBJECT-GROUP
    OBJECTS {
        jmQueueJobIndex, jmQueueNumberOfInterveningJobs, jmJobPriority,
        jmJobProcessAfterDateAndTime }
    STATUS  current
    DESCRIPTION
        "The queue group - conditionally mandatory."
    ::= { jmMIBGroups 2 }

jmCompletedGroup OBJECT-GROUP
    OBJECTS {
        jmCompletedJobIndex }
    STATUS  current
    DESCRIPTION
        "The completed group."
    ::= { jmMIBGroups 3 }

jmJobGroup OBJECT-GROUP
    OBJECTS {
        jmJobName, jmJobIdName, jmJobIdNumber, jmJobServiceTypes,
        jmJobOwner, jmJobDeviceNameOrQueueRequested, jmJobCurrentState,
        jmJobStateReasons }
    STATUS  current
    DESCRIPTION
        "The job group."
    ::= { jmMIBGroups 4 }

jmAttributeGroup OBJECT-GROUP
    OBJECTS {
        jmAttributeValueAsInteger, jmAttributeValueAsOctets }
    STATUS  current
    DESCRIPTION
        "The attribute group."
    ::= { jmMIBGroups 5 }


END
```

Appendix A - Mapping Of Job Submission Protocols To The Job Monitoring MIB Objects and Attributes

This appendix specifies the mapping of the input parameters of popular job submission protocols to the objects and attributes of the Job Monitoring MIB.

So far, this Appendix only has a few input parameters and only has ISO DPA. More input parameters will be added and more job submission protocols. The protocol list should include: **ISO DPA**, **Apple PAP**, **IPDS**, **LPR/LPD, NDPS, PJL, PostScript(tm), PSERVER, SMB, and IEEE 1284.1 (TIPSI)**. The Internet Printing Protocol (**IPP**) under development will be included as well.

Summary: the **jmJobIndex** is an Integer32(0..2147483647) data type and represents the job identifier attribute assigned by the server or device when the job is accepted by the server or device. The submitting user and client have no control over the value assigned by the server or device. The **jmJobIdName** and **jmJobIdNumber** are "client-side" identifiers that the submitting client specifies or is assigned by a downstream server on behalf of the client. The **jmJobIdName** is an alphanumeric OCTET STRING(SIZE(0..63)) one- or two-octet coded character set data type. The **jmJobIdNumber** is an Integer32(0..2147483647) data type.

Table 13-1 - Mapping of Job Submission Protocol Job Ids to the Corresponding MIB objects

Job Submission Protocol	jmJobIndex equiv. attribute	data type	jmJobIdName equiv. attribute	data type	jmJobIdNumber	data type
ISO DPA	job-identifier	ASCII(SIZE(0..4095))	job-client-id	OCTET STRING(SIZE(0..4095))	N/A	
LPD						
TBD...						

Appendix B - Comparison with ISO DPA

The ISO DPA attribute specifications have been moved from the JMP object specifications to this appendix for reference. The corresponding JMP object is indicated in the first column. If the second column is empty, there is no corresponding ISO DPA attribute.

14. Appendix B - Comparison with ISO DPA

The order of the groups is the same as the specification.

14.1 The General Group - comparison with ISO DPA

jmGeneralGroup (G)	Corresponding ISO DPA specification
1. jmJobSetIndex - a running index of Job Set instances supported by this device or server.	The client can get a list of jobs that are competing for a logical or physical printer that the client specifies as an input parameter.
2. jmGeneralJobSetName - The human readable administratively assigned name of this job set. Typically, this name will be the name of the job queue.	The logical printer or physical printer name.
3. **jmGeneralJobCompletedPolicy** -the time in seconds that jobs are kept in the **jmJobTable** and the **jmCompletedTable** after processing.	
4. **jmGeneralMaxNumberOfJobs** - the maximum number of job; -1 means no limit.	
5. jmGeneralNumberOfJobsToComplete - the total number of jobs currently in the Job Table that are to be completed.	
6. jmGeneralNumberOfJobsCompleted - the total number of jobs currently in the Job Table that are completed.	

1813 ## 14.2 The Queue Group - comparison with ISO DPA

jmQueueGroup (Q)	Corresponding ISO DPA specification
1. jmQueueIndex - a running index of the jobs that have *not* finished processing.	
2. **jmQueueJobIndex** - the job's identifier generated by the device or server implementing this Job Monitoring MIB	**Job-identifier** See below.
3. **jmQueueNumberOfInterveningJobs** - the number of jobs in front of this job	Intervening-jobs This attribute indicates the number of other jobs to be printed before this job may be scheduled for printing. The server shall set the value of this attribute to **0** when the job begins printing.
4. **jmJobPriority** - Job priority	Job-priority This attribute specifies a priority for scheduling the print-job. It is used by servers 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. The omission of this attribute implies that the user places no constraints concerning priority on the scheduling of the print-job.

jmQueueGroup (Q)	Corresponding ISO DPA specification
5. **jmJobProcessAfterDateAndTime** - The date and time after which the job shall become a candidate for processing.	Job-print-after This attribute specifies the calendar date and time of day after which the print-job shall become a candidate to be scheduled for printing. If the value of this attribute is in the future, the server shall set the value of the job's current-job-state to held and add the **job-print-after-specified** value to the job's **job-state-reasons** attribute and shall not schedule the print-job for printing until the specified date and time has passed. When the specified date and time arrives, the server shall remove the **job-print-after-specified** value from the job's **job-state-reason** attribute and, if no other reasons remain, shall change the job's current-job-state to pending so that the job becomes a candidate for being scheduled on printer(s). The server shall assign an empty value (see 9.1.2) to the job-print-after attribute when no print after time has been assigned, so that the job shall be a candidate for scheduling immediately.

1814

1815 14.3 The Completed Group - comparison with ISO DPA

jmCompletedGroup (C)	Corresponding ISO DPA specification
1. jmCompletedIndex - a running index of the jobs that have finished processing.	
2. **jmCompletedJobIndex** - the job's identifier generated by the device or server implementing this Job Monitoring MIB	**Job-identifier** See below.

1816 14.4 The Job Group - comparison with ISO DPA

jmJobGroup - Identification **(I)**	**Corresponding ISO DPA specification**
1. **jmJobIndex** - the job's identifier generated by the server or device implementing this Job Monitoring MIB	Job-identifier This attribute provides the job-identifier for this job on the server. The server shall generate a job-identifier value that is unique on that server, but need not be unique across the distributed environment. The value of the **job-identifier** attribute shall be returned by the server as part of the **PrintResult** in the first Print operation for the job (see 8.2.1). The client shall pass its value as part of the **PrintArgument** in subsequent Print operations for the same job.
2. **jmJobName** - Job name (assigned by job owner) which is not necessarily unique.	Job-name This attribute supplies a human readable string for the print-job. This string is used for naming the print-job in human-readable "free-form" fashion. 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 ListObjectAttributes result, or used in notification or logging messages. If this attribute is not specified, no job name is assumed, but implementation specific defaults are allowed, such as the value of the **document-name** attribute of the first document in the job.

jmJobGroup - Identification **(I)**	**Corresponding ISO DPA specification**
3. jmJobIdName - the job's identifier name generated by the job submitting software using the job submission protocol. This name can be anything that helps identifier the job to the job submitter, including the name of the queue from which the job was submitted.	**Job-client-id** This attribute supplies a human-readable descriptor for the job. This descriptor may be printed by the server on auxiliary sheets to help identify the user's printed output, and discriminate between different jobs. Use and treatment of this attribute is implementation and site specific. If the client specifies the value of the job attribute **job-client-id**, no server shall change it. If the client does not specify the value of the job attribute **job-client-id**, the first server shall set it to the value of the job attribute **job-identifier**, so that no downstream server shall change it. These rules ensure that if an implementation prints the value of the **job-client-id** on an auxiliary sheet, it has a value that is meaningful to the client originally submitting the job, no matter how many servers the job passes through. For example, client A submits a job to server B and does not specify a value for the job attribute **job-client-id**. Server B assigns a **job-identifier** of **123** to the job, and forwards this job to server C. Server C assigns a **job-identifier** of **456** to the job and forwards this job to printer D. Printer D is not a DPA server, but it has its own queue and assigns a job-id of **789** to the job. The following table shows the value of the relevant job attributes in the two servers B and C:
4. jmJobIdNumber - the job's identifier number generated by the job submitting software using the job submission protocol. A (-2) value shall indicate that the submitter did not supply a job identifier number.	
5. **jmJobServiceTypes** - Job types (print, fax, scan, etc.) - bit vector to get multiple values in a single object	

jmJobGroup - Identification **(I)**	**Corresponding ISO DPA specification**
6. **jmJobOwner** - Job owner (User name of the user that originally submitted the job)	Job-owner This attribute supplies the name of the human owner of the print-job, i.e., the name of the user who submitted the job originally, not the user who most recently (re)submitted the job. The value of **job-owner** will often be the same as **job-originator**. The **job-owner** will be different from **job-originator** when the job has been submitted by the originator on behalf of the owner. This attribute is not to take the place of the security parameters or the access-and-accounting attributes. If this attribute is not specified, the value of **user-name** or **job-originator** should be used for any circumstances which require a value for **job-owner**.
7. **jmJobDeviceNameOrQueueRequested** - Device name (Device-specific name of device) or queue requested by the submitting user.	Printer-name-requested This attribute identifies the printer to be used for printing the job. The client shall specify the value of this attribute with the first invocation of the **Print** operation for the print-job as the explicit **printer-name** component of the **PrintArgument**, rather than as an attribute (see 8.2.1.1). NOTES 1 To cause a server to select a printer according to other attributes, the system administrator should define a logical printer that supports ALL of the physical printers supported by the server. 2 For the server that supports only a single printer, the logical printer name may be the same as the server name, as long as they cannot be confused for each other in the name service directory. 3 Initial-value-job objects should have the value of their **printer-name-requested** attribute specified as an empty value in order to indicate that no printer-name is defaulted.

jmJobGroup - Status (S)	**Corresponding ISO DPA specification**	
14. **jmJobCurrentState** - Job state (**pending**, **processing**, **completed**, etc.)	Current-job-state This attribute identifies the current state of the job (pending, printing, held, etc.) The following job state standard values are defined:	
	Descriptive Name	Descriptor Text
	unknown	The job state is not known, or is indeterminate.
	pre-processing	The job has been created on the server by the **create-job** sub-operation of the print-request, but a print-request with a **TRUE** value for the **job-submission-complete** component of the PrintArgument has not yet been received and no document has started processing. The job maybe in the process of being checked by the server for attributes, defaults being applied, a printer being selected, etc.
	held	The job is waiting to be released for scheduling for any number of reasons as specified by the value of the job's **job-state-reasons** attribute.
	pending	The job's **job-submission-complete** attribute is **TRUE** since the server has received a print-request with the **job-submission-complete** parameter **TRUE** and the job is waiting to start processing on a printer.
	processing	The server is processing the job, or 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.
	paused	The job has been paused as a result of a PauseJob operation.
	interrupted	The job was interrupted by the InterruptJob request for an intervening job, and shall resume processing automatically once the intervening job has completed.
	terminating	The job has been cancelled by a CancelJob request or aborted by the server and is in the process of terminating. The job's **job-state-reasons** attribute contains the reasons that the job is being terminated.

jmJobGroup - Status (S)	**Corresponding ISO DPA specification**	
	retained	The job is being retained at the server as a result of the job's **job-retention-period** being non-zero. The job has (1) completed successfully or with warnings or errors, (2) been aborted while printing by the server, or (3) been cancelled by the CancelJob request before or during processing. The job's **job-state-reasons** attribute contains the reasons that the job has been retained. While in the **retained** state, all of the job's document data (and resources, if any) shall be retained by the server; thus a job in the **retained** state could be reprinted, using some means outside the scope of ISO\IEC 10175-Part 1.

jmJobGroup - Status (S)	**Corresponding ISO DPA specification**	
	completed	The job has: (1) completed successfully or with warnings or errors, (2) been aborted by the server while printing, or (3) been cancelled by the **CancelJob** request, AND the job's: (1) **job-retention-period** was zero or has expired, or (2) **job-discard-time** has arrived. The job's **job-state-reasons** attribute contains the reason(s) that the job has been completed. While in the **completed** state, a job's document data (and resources if any) need not be retained by the server; thus a job in the **completed** state could not be reprinted. The length of time that a job may be in this state, before transitioning to **unknown**, is implementation-dependent. However, servers that implement the **completed** job-state shall retain, as a minimum, the following attributes for any job in the completed state: **job-identifier**, **job-owner**, **job-name**, **current-job-state**, **printers-assigned**, and **job-state-reasons**. Print clients and DP-Servers shall be prepared to receive all the standard job states. DP-Servers are not required to generate all job states, only those which are appropriate for the particular implementation. If a server implementation or policy is to start processing documents before the last print-request (with a **TRUE** value for the **job-submission-complete** parameter) and the value of the job's **job-scheduling** attribute is not **after-complete**, the server shall change the job's **current-job-state** from **pre-processing** directly to the **processing** state when the server begins processing any of the job's documents.

jmJobGroup - Status (S)	**Corresponding ISO DPA specification**
15. **jmJobStateReasons** - Job state reasons - additional information about the job state: reasons being held, additional completed information such as successful, warnings, or errors.	Job-state-reasons This attribute identifies the reason or reasons that the job is in the **held, terminating**, **retained**, or **completed** state. The server shall indicate the particular reason(s) by setting the value of the **job-state-reasons** attribute. When the job is not in any of these states, the server shall set the value of the **job-state-reasons** attribute to the empty set. The following [DPA] standard values are defined: **documents-needed, job-hold-set, job-print-after-specified, required-resources-not-ready, successful completion, completed-with-warnings, completed-with-errors, cancelled-by-user, cancelled-by-operator, aborted-by-system, logfile-pending ,** and **logfile-transferring.**

1819 **14.5 The Attribute Group - comparison with ISO DPA**

jmAttributeGroup (R)	Corresponding ISO DPA specification
1. **jmJobIndex** - the job's current identifier generated by the server or device implementing this Job Monitoring MIB	**job-identifier** See above.
2. **jmAttributeTypeIndex** - identifies which attribute is being represented by this row:	Corresponds to the attribute-type OID that identifies each attribute in ISO DPA.
a) other(1) - not one of the following	
b) fileName(3) - file name of the document.	**Document-file-name** This attribute specifies the file name of the document, if the document came from a file. The file name may include the full path to the file, in which case the **name-syntax** element of the **DistinguishedNameString** data type shall specify the syntax of the file name. If the document did not come from a file, the client should not specify this attribute.
c) **documentName(4) -** Document name (defaults from the file-name)	**Document-name** This attribute supplies a human readable string for the document. This string is used for naming the document in a human-readable "free-form" fashion. This attribute is intended for enabling a user or the user's application to convey a document name that may be printed on a start sheet, returned in a ListObjectAttributes result, or used in notification or logging messages. If this attribute is not specified, no document name is assumed, but implementation specific defaults are allowed, such as the simple-name part of the value of the **document-file-name** attribute. It is suggested, however, that the server not supply additional text for this attribute when printing its value (e.g. on a start sheet). This string only has meaning to the clients and can therefore take several forms, e.g. the name of a mail folder, name of a revisable document, the file specification minus the file path, the title of a document, etc.

jmAttributeGroup (R)	Corresponding ISO DPA specification
d) jobAccountName(5) - name of the account to which the job shall be charged.	**Accounting-information** This attribute specifies information required by accounting services (e.g. the account to be charged for any services rendered). Accounting information is intended to be interpreted by an accounting system, and may be opaque to the print service.
e) jobComment(6) - free form comment.	**Job-comment** This attribute supplies an arbitrary human-readable text string associated with the print-job. This attribute is intended for enabling a user to convey a text string that may be printed on a job start sheet, for example, in an implementation-dependent manner.
f) processingMessage(7) - current job status and any problems as a human readable message.	
g) jobSourceChannelIndex(8) - index in Printer MIB of the job source channel.	

jmAttributeGroup (R)	Corresponding ISO DPA specification
h) outputBinIndex(9) - index in the Printer MIB of the output bin(s) that this job is using.	**results-profile.output-bin** The **output-bin** element specifies the output receptacle for the media on which the job-result-set is to be printed. The **NameOrOid** type provides two choice types for use in system implementations that (1) use a simple-named bin identification and (2) for those that use named bins that are identified with object identifiers. The **output-bin** element specifies the output receptacle for the media on which the job-result-set is to be printed. The **NameOrOid** type provides two choice types for use in system implementations that (1) use a simple-named bin identification (which may consist of a simple-name or solely of numeric digits for numbered bins, including leading 0 digits), and (2) for those that use named bins that are identified with object identifiers. The correspondence between the integer name of an output-bin and the actual output-bin in the printer is printer-dependent, and an output-bin named by a simple-name may also have an object identifier that names the output-bin as well. A server may try to convert a simple-name received from a client to one of the server's OIDs, depending on implementation. However, a server shall always return an output-bin as an OID to the client if the server identifies the output-bin using an OID.
i) outputBinName(10) - name of the output bin(s) that the job is using.	**results-profile.output-bin** See above.
j) **sides(11) -** Number of sides requested (one-sided, two-sided)	**Sides** This attribute specifies the number of printable surfaces of the medium to be imaged.

jmAttributeGroup (R)	Corresponding ISO DPA specification
k) **documentFormatIndex(12) -** the index in the Printer MIB of the interpreter(s) that the job requires/uses.	**Document-format** This attribute identifies the overall print document format used for the document. It consists of three elements, a **document-format**, a **document-format-variants** and a **document-format-version**. The latter two elements are optional. The **document-format** element identifies a particular family of document formats, of which there may exist several versions or variants. The **document-format-variants** and **document-format-version** elements identify a specific instance of a document format. The variant refers to a particular functional subset of a format. For example, the format PostScript has variants of level 1 and level 2, and the format PCL has several variants, including PCL4 and PCL5. The version distinguishes among successive releases of the same basic format and variant. For example, successive versions of Xerox Interpress include versions 2.0, 2.1, 3.0, 3.1, etc. Put in a separate table so can have multiple values, one for each document.
l) **documentFormatEnum(13) -** the enum identifying the interpreter(s) that the job requires/uses.	**document-format** See above.

jmAttributeGroup (R)	Corresponding ISO DPA specification

m) **physicalDevice(14) -** physical devices used	**printers-assigned**
	
	This attribute identifies the physical printer or printers to which this job has been assigned, if any.
	
	When the job is first submitted and the server has not yet assigned any printers to the job, the **SEQUENCE** shall be empty.
	
	If the server intends to use a single printer for the job, and the server has assigned a printer to the job, the **SEQUENCE** shall contain just that printer.
	
	If a server has split the job into multiple pieces and assigned each piece to a different printer, the **SEQUENCE** shall contain n elements, one for each assigned printer. A job with multiple job-result-sets is an example of a job that would be easy to split into multiple pieces.
	
	`printers-assigned ATTRIBUTE`
	`        WITH ATTRIBUTE-SYNTAX`
	`distinguishedNameStringSequenceSyntax`
	`        SINGLE VALUE`
	`        ::= id-att-printers-assigned`
	
	A **SEQUENCE** with no elements shall be returned if this attribute is supported, but this job has not yet been assigned to any physical printers.
	
	The number of elements in the **SEQUENCE** for this attribute shall be the same as the number of elements in the **SEQUENCE** for the associated job attribute **printer-state-of-printers-assigned**.
	
	In addition, the *i*th element of the value of **printer-state-of-printers-assigned** shall be the state of the printer named by the *i*th element of **printers-assigned**.
	
	The p**rinters-assigned** value shall not be the same as the printer requested by the user if the job's **printer-name-requested** attribute specified a logical printer that supports one or more different physical printers. The **printers-assigned** value might differ also if the job has been re-assigned by an operator to ensure successful completion of the job, allowing the user to find out where a job has been re-assigned (when necessary).
	
	The value of the job's **printers-assigned** attribute shall remain after the job has completed, so that users can determine the physical printer(s) on which the job was printed.

jmAttributeGroup (R)	Corresponding ISO DPA specification
n) physicalDeviceName(15) - the physical device name(s) used or being used by the job.	**printers-assigned** See above.
o) **jobCopiesRequested(16) -** Number of job copies requested	**job-copies** Total number of job copies in the job, i.e., number of job copies summed across the job-result-sets. Whether job copies are collated or not depends on implementation. NOTE - In ISO DPA, job-copies is a separate value for each job result set, not the summation. But it didn't seem worth the effort to make job-copies a table for the MIB.
p) **jobCopiesCompleted(17) -** Number of job copies produced	**total-job-copies** Total number of job copies in the job, i.e., number of job copies summed across the job-result-sets. Whether job copies are collated or not depends on implementation. NOTE - In ISO DPA, job-copies is a separate value for each job result set, not the summation. But it didn't seem worth the effort to make job-copies a table for the MIB.
q) **documentCopiesRequested(18) -** Number of document copies requested	**copy-count** This attribute specifies the number of copies of the documents, or of the selected pages of the document, to be printed. In ISO DPA, there is a **copy-count** attribute for each document in the job. The proposal here is to have a single per-job count of the number of copies of documents, in order to avoid a per-document table.
r) **documentCopiesCompleted(19) -** Number of document copies completed	**copies-completed** In ISO DPA, there is a **copy-count** attribute for each document in the job. The proposal here is to have a single per-job count of the number of copies of documents, in order to avoid a per-document table.

jmAttributeGroup (R)	Corresponding ISO DPA specification
s) jobKOctetsTotal (20)- total K octets to be processed in the job - rounded up to next higher K (1024)	**total-job-octets** This attribute indicates the size of the job in octets, including document and job copies. total-job-octets ATTRIBUTE WITH ATTRIBUTE-SYNTAX cardinal64Syntax SINGLE VALUE ::= id-att-total-job-octets The server may update the value of this attribute after each document has been transferred to the server or the server may provide this value after all documents have been transferred to the server, depending on implementation. In other words, while the job is in the **pre-processing** state and when the job is in the **held** state with the **job-state-reasons** containing a **document-needed value**, the value of the **total-job-octets** job status attribute depends on implementation and may not correctly reflect the size of the job. In computing this value, the server shall include the multiplicative factors contributed by the (1) **copy-count** document attribute, (2) the **results-profile**.**job-copies** job attribute element and (3) multiple values of the **results-profile** job attribute, independent of whether the printer can process multiple copies of the job or document without making multiple passes over the job or document data and independent of the value of the **output** document attribute (**page-collate** vs. **no-page-collate**). Thus the server computation is independent of the printer implementation and shall be: 1. Document contribution: Multiply each **copy-count** by the size of the document in octets. 2. Add each document contribution together 3. Job result contribution: Multiply the job size by the number **job-copies** in the result set. 4. Add each job result contribution together Multiply the value by the number of values in the job's **result-profile** attribute.

jmAttributeGroup (R)	Corresponding ISO DPA specification
t) **jobKOctetsCompleted(21) -** K octets completed - rounded up to nearest K (1024).	**Octets-completed** This attribute indicates the number of octets of the job that the printer(s) have completed printing. The server shall not reset its value during the processing of multiple copies of documents or the job. Since this attribute is intended to measure the progress of a job, the value shall include repeated pages due to multiple copies. The accuracy of this value is implementation-dependent. It may be approximated by the number of octets conveyed to the printer. This attribute may not be supported for all printers and all page description languages. The value of this attribute shall be **0** if printing has not started for this job.
u) impressionsSpooled(22) - impressions spooled for the job.	
v) impressionsSentToDevice(23) - impressions sent to the device for the job.	
w) impressionsInterpreted(24) - impressions interpreted for the job.	

jmAttributeGroup (R)	Corresponding ISO DPA specification
x) **impressionsRequested(25)** - impressions completed	**job-impression-count** This attribute contains the number of impressions that the server expects the printer to make. The server shall compute this value by the following procedure: a) For each document in the job object, multiply the value of document's **page-count** attribute by the value of its **copy-count** attribute. Then divide the result by the value of **number-up** (if non-zero) and make into an integer using the ceiling operator. Call the result *document-set-impression-count*. NOTE – The **number-up** attribute may contain a number or an OID. For the OID case, the server either knows implicitly what number is associated with the OID or it must query the **number-up** object for its **imposition-n-up** attribute. In the case where the server cannot obtain the value, it should assume the value of **number-up** is 1. b) Add up all the *document-set-impression-counts* from the previous step and call this sum the *job-copy-impression-count*. c) For each job-result-set, multiply the value of *job-copy-impression-count* from the previous step by the value of **job-copies** element of the job-result-set and call the result *job-result-set-impression-count*. d) Add up all the *job-result-set-impression-counts* from the previous step and set this sum into the **job-impression-count** attribute. The value of this attribute is a measure of the amount of time the job will take to print on printers with a single print engine. The accuracy of this value is dependent on the accuracy of the **page-count** attribute in each document. If some documents have a **page-count** value of 0, the server may set the value of this attribute to 0 and not use it for scheduling.

jmAttributeGroup (R)	Corresponding ISO DPA specification
y) **impressionsCompleted(26) -** impressions completed for the job.	**impressions-completed** This attribute indicates the number of impressions that the printer engine(s) have placed on the media for the job. See the note in the **pages-completed** attribute for the relationship of the **pages-completed**, **impressions-completed** and **media-sheets-completed** attributes. The server shall not reset its value during the processing of multiple copies of documents or the job. Since this attribute is intended to measure the progress of a job, the value shall include repeated pages due to multiple copies. When the job completes, this attribute should contain the value of the total number of impressions that the printer made for the print-job. The accuracy of this value is implementation-dependent. It is expected that the value reported is never greater than the actual value. This attribute may not be supported for all printers and all page description languages. The value of this attribute shall be 0 if printing has not started for this job.
z) **impressionsCompletedCurrentCopy(27) -** impressions completed on the current copy.	

jmAttributeGroup (R)	Corresponding ISO DPA specification
aa) pagesRequested(28) - logical pages requested to be processed	**job-page-count** This attribute contains the number of source pages in the job that the server expects to image. The server shall compute this value by the following procedure: a) For each document in the job object, multiply the value of document's **page-count** attribute by the value of its **copy-count** attribute and call the result *document-set-page-count*. b) Add up all the *document-set-page-counts* from the previous step and call this sum the *job-copy-page-count*. c) For each job-result-set, multiply the value of *job-copy-page-count* from the previous step by the value of **job-copies** element of the job-result-set and call the result *job-result-set-page-count*. d) Add up all the *job-result-set-page-counts* from the previous step and set this sum into the **job-page-count** attribute. The value of this attribute is a measure of the amount of computation involved. The accuracy of this value is dependent on the accuracy of the **page-count** attribute in each document. If some documents have a **page-count** value of 0, the server may set the value of this attribute to 0 and not use it for scheduling.

jmAttributeGroup (R)	Corresponding ISO DPA specification
bb)pagesCompleted(29) - logical pages completed for the job.	**pages-completed** This attribute indicates the number of pages of the job that the printer(s) have completed printing. NOTE – The number of source pages, impressions and sheets of media may differ. The following examples illustrate how they may differ when attributes, rather than the document contents, control the printing. If **number-up** is 0 or 1, there is one source page per impression, and if **number-up** is 2, there are two source pages per impression. If **sides** is 1, there is one impression per sheet of media, but if **sides** is 2, there are two impressions per sheet of media. By inference, if **number-up** is 4 and **sides** is 2, there are 4 source pages per impression and 8 source pages per sheet of media. The server shall not reset its value during the processing of multiple copies of documents or the job. Since this attribute is intended to measure the progress of a job, the value shall include repeated pages due to multiple copies. When the job completes, this attribute should contain the value of the total number of source pages that the printer processed for the print-job. The accuracy of this value is implementation-dependent. It is expected that the value reported is never greater than the actual value. This attribute may not be supported for all printers and all page description languages. The value of this attribute shall be **0** if printing has not started for this job.
cc) pagesCompletedCurrentCopy(30) - logical pages completed on the current copy.	

jmAttributeGroup (R)	Corresponding ISO DPA specification
dd)**sheetsRequested(31) -** sheets requested to be processed.	**job-media-sheet-count** This attribute contains the number of sheets of media that the server expects to consume for the job. The server shall compute this value by the following procedure: a) For each document in the job object, multiply the value of document's **page-count** attribute by the value of its **copy-count** attribute. Then divide the result by the value of **number-up** (if non-zero) and make into an integer using the ceiling operator. Then, if **sides** is 2, divide the result by 2 and round. Call the result *document-set-media-sheet-count*. NOTE – See the note on **number-up** in the **job-impression-count** attribute. b) Add up all the *document-set-media-sheet-counts* from the previous step and call this sum the *job-copy-media-sheet-count*. c) For each job-result-set, multiply the value of *job-copy-media-sheet-count* from the previous step by the value of **job-copies** element of the job-result-set and call the result **job-result-set-media-sheet-count**. d) Add up all the *job-result-set-media-sheet-counts* from the previous step and set this sum into the **job-media-sheet-count** attribute. The value of this attribute is a measure of the total number of sheets of media that will be consumed and it is a good measure of the amount of time the job will take to print on printers with two print engines, one for each side of the media. The accuracy of this value is dependent on the accuracy of the **page-count** attribute in each document. If some documents have a **page-count** value of 0, the server may set the value of this attribute to 0 and not use it for scheduling.

jmAttributeGroup (R)	Corresponding ISO DPA specification
ee) **sheetsCompleted(32) -** sheets completed for the job.	This attribute indicates the number of sheets of media that the printer(s) have completed printing for the job. See the note in the **pages-completed** attribute for the relationship of the **pages-completed**, **impressions-completed** and **media-sheets-completed** attributes. The server shall not reset its value during the processing of multiple copies of documents or the job. Since this attribute is intended to measure the progress of a job, the value shall include repeated pages due to multiple copies. When the job completes, this attribute should contain the value of the total number of sheets of media used for the print-job. The accuracy of this value is implementation-dependent. It is expected that the value reported is never greater than the actual value. This attribute may not be supported for all printers and all page description languages. The value of this attribute shall be 0 if printing has not started for this job.
ff) **sheetsCompletedCurrentCopy(33) -** sheets completed on the current copy.	
gg) **mediumRequested(34) -** the medium(a) requested for this job, kind and number.	
hh) **mediumConsumed(35) -** the medium(a) consumed for this job, kind and number.	
ii) **colorantRequestedIndex(36)**	
jj) **colorantRequestedName(37)**	
kk) **colorantConsumedIndex(38)**	
ll) **colorantConsumedName(39)**	
mm) **jobSubmissionDateAndTime(40)**	**Submission-time** This attribute indicates the time at which the latest print request for this job was accepted by the server.

jmAttributeGroup (R)	Corresponding ISO DPA specification
nn) **jobSubmissionTimeStamp (41)**	**Submission-time** See above.
oo) **jobStartedProcessingDateAndTime(42)**	**started-printing-time** This attribute indicates the time at which this job started printing.
pp) **jobStartedProcessingTimeStamp(43)**	**started-printing-time** See above.
qq) **jobCompletedDateAndTime(44)**	**completion-time** This attribute indicates the time at which this job completed. Providing this time is useful for jobs which are retained after printing.
rr) **jobCompletedTimeStamp(45)**	**completion-time** See above.
ss) **processingCPUTime(46) -** Processing time so far, not counting needs attention time.	**processing-time** This attribute indicates how long an individual job has been processing [in seconds].
3. **jmAttributeInstanceIndex-** attribute instance index for the job as a whole or document number if an attribute is per-document.	ISO DPA has multi-valued job attributes and as per-document attributes.
4. jmAttributeValueAsInteger- attribute value as an integer.	
5. jmAttributeValueAsOctets- attribute value as an OCTET STRING for coded characters (text) or binary bit strings or binary octet strings.	

15. APPENDIX C - Comparison of Job Submission Protocols to JMP Objects

The JMP objects and attributes are divided into the following categories:

1. Job Identification (I)
2. Job Parameters (P)
3. Job Status and Accounting (S)

The following table lists each JMP object and attribute and indicates in each column whether there is a corresponding input parameter in the indicated job submission protocol.

The first column contains the MIB name followed by a descriptive name for the object.

The **Conf**. column specifies the conformance:

M means **Mandatory** for conformance to this MIB specification

CM means **Conditional** Mandatory (for spooling systems, and systems with day and time clocks, etc.).

The **Cardinality** columns contains:

1 meaning there is only **one** of these objects per job, so that the object can be in a table that is indexed by **jmJobSetIndex** and **jmJobIndex**.

n meaning that there may be **more than one** of these objects per job, so that that the object must be in another table that in indexed by **jmJobSetIndex**, **jmJobIndex**, and **jmAttributeInstanceIndex**

Job Identification (I)	**Con for man ce**	**Car dina lity**	**IS O DP A**	**Ap ple PA P**	**IP DS**	**LP R/ LP D**	**ND PS**	**PJ L**	**PSE RV ER**	**S M B**	**TIP SI**
jmQueueNumberOfInterveningJobs - the number of jobs in front of this job											
jmJobPriority - Job priority: 1 to 100.	**CM**	**1**	**x**				**x**				***x***
jmJobProcessAfterDateAndTime - date and time after which the job becomes a candidate for processing	**CM**	**1**	**x**								
jmJobIndex - Job current id generated by the server implementing this Job Monitoring MIB when the job was submitted)	**M**	**1**	**x**		**x**	**x**	**x**	**x**		**x**	
jmJobName - Job name assigned by job owner which is not necessarily unique.	**M**	**1**	**x**		**x**		**x**	**x**	**x**		
jmJobIdName - the job's identifier name generated by the job submitting software using the job submission protocol. This name can be anything that helps identifier the job to the job submitter, including the name of the queue from which the job was submitted.	**M**	**1**	**x**	**x**		**x**	**x**		**x**	**x**	***x***
jmJobIdNumber - the job's identifier number generated by the job submitting software using the job submission protocol. A (-2) value shall indicate that the submitter did not supply a job identifier number in the job submission protocol.	**M**	**1**									
jmJobServiceTypes - Job types (print, fax, scan, etc.) - bit vector to get multiple values in a single object	**M**	**1**			**x**		**x**			**x**	
jmJobOwner - Job owner (User name of the user that originally submitted the job)	**M**	**1**	**x**	**x**	**x**		**x**		**x**	**x**	***x***
jmJobDeviceNameOrQueueRequested - Device name (Device-specific name of device) or queue name requested by the submitting user.	**M**	**1**	**x**		**x**		**x**				**x**

Job Status (S)	**Conformance**	**Cardinality**	**ISO DPA**	**Apple PAP**	**IPDS**	**LPR/ LPD**	**NDPS**	**PJL**	**PSERVER**	**SMB**	**TIPSI**
1. **jmJobCurrentState** - Job state (held, pending, processing, completed, etc.)	**M**	**1**	**x**	**x**		**x**	**x**	**x**		**x**	**x**
2. **jmJobStateReasons** - Job state reasons - additional information about the job state: reasons being held, additional executing information such as device(s) needs attention, additional completed information such as successful, warnings, or errors.	**M**	**1**	**x**		**x**		**x**	**x**			**x**
3. **jmAttributeTypeIndex** - Attributes representing information and resources required/consumed (table):	**M**	**n**									
a) Other											
b) File names	**CM**	**n**	**x**								
c) Document name(s) (or file-names)	**CM**	**n**	**x**	**x**	**x**	**x**	**x**		**x**		**x**
d) **jobAccountName** - Account Name	**CM**	**1**	**x**				**x**				**x**
e) **jobComment** - Job comment	**CM**	**1**	**x**				**x**	**x**	**x**		***x***
f) **processingMessage(7)**	**CM**	**n**									
g) **jobSourceChannelIndex** - Source channel (index of channel row in Printer MIB)	**CM**	**1**		**x**		**x**					**x**
h) **outputBinIndex**(9)	**CM**	**n**									
i) **outputBinName**(10)	**CM**	**n**	**x**								
j) Number of sides requested (one-sided, two-sided)	**CM**	**1**	**x**		**x**		**x**	**x**			***x***
k) PDLs requested/used - index	**CM**	**n**									
l) PDL requested/used - enum	**CM**	**n**	**x**			**x**	**x**	**x**			**x**

Job Status (S)	**Conformance**	**Cardinality**	**ISODPA**	**AppleP AP**	**IPDS**	**LPR/LPD**	**NDPS**	**PJL**	**PSERVER**	**SMB**	**TIPSI**
m) **jmDeviceIndex(14)** - the host resources index of the corresponding Printer MIB that the job was submitted to or has been assigned to be printed on by the server. 0 indicates if the server has not assigned a printer to the job.	**CM**	**n**									
n) **physicalDeviceName(15) -** the physical device name(s) used or being used by the job.	**CM**	**n**	**x**		**x**		**x**	**x**	**x**		**x**
o) Number of job copies requested	**CM**	**1**	**x**				**x**	**x**	**x**		
p) Number of job copies completed	**CM**	**1**	**x**								
q) Number of document copies requested	**CM**	**1**	**x**				**x**	**x**	**x**		
r) Number of document copies completed	**CM**	**1**	**x**								
s) jobKOctetsTotal - total K octets to be processed in the job - rounded up to next K value.	**CM**	**1**	**x**								
t) **jobKOctetsCompleted** - K octets completed - should be rounded down to lower K until completed.	**CM**	**1**	**x**				**x**				**x**
u) impressionsSpooled(22) - impressions spooled for the job.	**CM**	**1**									
v) impressionsSentToDevice(23) - impressions sent to the device for the job.	**CM**	**1**									
w) impressionsInterpreted(24) - impressions interpreted for the job.	**CM**	**1**									
x) impressionsRequested(25) - impressions requested	**CM**	**1**									

Job Status (S)	Conformance	Cardinality	ISO DPA	Apple PAP	IPDS	LPR/LPD	NDPS	PJL	PSERVER	SMB	TIPSI
y) **impressionsCompleted(26) -** impressions (sides) completed for the job.	**CM**	**1**	**x**				**x**	**x**			
z) **impressionsCompletedCurrentCopy(27) -** impressions completed on the current copy.	**CM**	**1**									
aa) **pagesRequested(28) -** logical pages requested to be processed	**CM**	**1**									
bb) **pagesCompleted(29) -** logical pages completed for the job.	**CM**	**1**	**x**								
cc) **pagesCompletedCurrentCopy(30) -** logical pages completed on the current copy.	**CM**	**1**	**x**								
dd) **sheetsRequested(31) -** sheets requested to be processed.	**CM**	**1**									
ee) **sheetsCompleted(32) -** sheets completed for the job.	**M**	**1**	**x**				**x**				
ff) **sheetsCompletedCurrentCopy(33) -** sheets completed on the current copy.	**CM**	**1**									
gg) **mediumRequested(34)** - the medium(a) requested for this job, kind and number.	**CM**	**n**									
hh) **mediumConsumed(35) -** the medium(a) consumed for this job, kind and number.	**CM**	**n**									
ii) **colorantRequestedIndex(36)**	**CM**										
jj) **colorantRequestedName(37)**	**CM**	**n**									
kk) **colorantConsumedIndex(38)**	**CM**	**n**									
ll) **colorantConsumedName(39)**	**CM**	**n**									

Job Status (S)	**Conformance**	**Cardinality**	**ISO DPA**	**Apple PAP**	**IPDS**	**LPR/LPD**	**NDPS**	**PJL**	**PSERVER**	**SMB**	**TIPSI**
mm) **jmJobSubmissionDateAndTime** - Date/Time of job submission by job owner	**CM**	**1**	**x**				**x**		**x**	**x**	
nn) jobSubmissionTimeStamp(41)	**CM**	**1**									
oo) **jobStartedProcessingDateAndTime** - Date/Time of day job started processing on device	**CM**	**1**	**x**				**x**				**x**
pp) jobStartedProcessingTimeStamp(43)	**CM**	**1**									
qq) **jobCompletionDateAndTime** - Date/Time of day job finished using the device	**CM**	**1**	**x**								
rr) jobCompletedTimeStamp(45)	**CM**	**1**									
ss) Processing CPU time so far	**CM**	**1**	**x**				**x**				
8. **jmAttributeValueAsInteger** - attribute as integer value	**M**	**n**									
9. jmAttributeValueAsOctets - attribute value as coded character data or octet string.	**M**	**n**									

Appendix D - Use of MS-WORD Version 6.0 to format the MIB

16. Appendix D - Use of MS-WORD Version 6.0 to format the MIB

This appendix describes how this MIB specification was created using MS-WORD to perform the formatting and produce plain text, 72-columns wide, with only ASCII characters, and running headers and footers as required by the IETF RFCs and Internet Drafts.

Don't use smart quotes. To turn off: Tools/AutoCorrect/ replace straight quotes with smart quotes, turn off.

The word template mib.dot was created with the following styles:

1. **Fixed** - CourierNew 12 point set which gives 10 characters per inch. Also set line spacing exactly 12 point. Have no leading indent. Have no right indent. Depend on the margins to wrap whether on full lines or in tables.
2. **Fixed Indent** - indents 4 characters (0.4 inches)
3. **Fixed Double Indent** - indents 8 characters (0.8 inches)
4. **Comment Full** - full line comments.
5. **Quoted Running Text** - indented 8 characters
6. **Normal** - TimesRoman 12 point for text that is outside the BEGIN END statements while reviewing the document. To produce the Internet Draft, change the definition of the Normal style to use the Courier 12 point with line spacing exactly 12 point.

The following macros are defined in mib.dot with speed keys indicated in parens:

1. **CreateFullComment (ALT+C) -** creates a full line comment as two column table with the first column being 3 characters wide for the ASN.1 "-- "comment characters. The second column is the full line comments with line wrapping.
2. **CreateMIBGroup (ALT+G) -** produces a skeleton group to be filled in.
3. **CreateMIBObject (ALT+O) -** produces a skeleton OBJECT-TYPE to be filled in
4. **CreateTC (ALT+T) -** produces a skeleton textual-convention to be filled in.

To produce the final plain text, follow the following steps:

1. Accept all revisions
2. Redefine **Normal** style to be CourierNew 12 point with exactly 12 point line spacing.
3. Set the left and right margins to 0 and 1.3, so that text comes out without leading spaces and has exactly 72 characters (8.5-1.3=7.2).
4. Set the top and bottom margins to 0.
5. Select the entire document and type Control Q to get rid of all character formatting, such as bold, italic, etc. Since all indents were done with styles, no indention changes. (be sure not to use the toolbar to indent, else the Control Q will undo that).
6. Replace the table of contents (since new pagination) and make sure NOT to have any leader for the table of contents, figure table, or table of issues. Else the generic text driver will output CR with overstrike which won't meet IETF requirements for plain text.
7. Select the generic text printer (but do not keep selected, else always get fixed pitch font, no matter what font selected).
8. Output to file. This will produce a file with headers and footers that meet IETF requirements.

17. Author's Addresses

Ron Bergman
Dataproducts Corp.

Phone: 805-578-4421
Fax:
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

Phone: 801-861-7366
Fax: 801-861-4025
EMail: scott_isaacson@novell.com

Harry Lewis
IBM Corporation
P.O. Box 1900
Boulder, CO 80301-9191

Phone: (303) 924-5337
Fax:
Email: harryl@vnet.ibm.com

Send comments to:
JMP Mailing List: jmp@pwg.org

JMP Mailing List Subscription Information:
jmp-request@pwg.org

Other Participants:

Chuck Adams - Tektronix
Jeff Barnett - IBM
Keith Carter, IBM Corporation
Jeff Copeland - QMS
Andy Davidson - Tektronix
Roger deBry - IBM
Mabry Dozier - QMS
Lee Ferrel - Canon
Steve Gebert - IBM
Robert Herriot - Sun Microsystems Inc.
Shige Kanemitsu - Kyocera
David Kellerman - Northlake Software
Rick Landau - Digital
Harry Lewis - IBM
Pete Loya - HP
Ray Lutz - Cognisys
Jay Martin - Underscore
Mike MacKay, Novell, Inc.
Stan McConnell - Xerox
Carl-Uno Manros, Xerox, Corp.
Pat Nogay - IBM
Bob Pentecost - HP
Rob Rhoads - Intel
David Roach - Unisys
Hiroyuki Sato - Canon
Bob Setterbo - Adobe
Gail Songer, EFI
Mike Timperman - Lexmark
Randy Turner - Sharp
William Wagner - Digital Products
Jim Walker - Dazel
Chris Wellens - Interworking Labs
Rob Whittle - Novell
Don Wright - Lexmark
Lloyd Young - Lexmark
Atsushi Yuki - Kyocera
Peter Zehler, Xerox, Corp.

18. Change History (not to be included in the Internet Draft)

All future changes will be recorded here in *reverse* chronological order by version.

18.1 Changes to version 0.7, dated 3/13/97 to make version 0.71, dated 3/26/97

1. Made the formatting changes necessary to make an Internet Draft.
2. Replaced Figure 1 with a Job State Transition table.
3. Clarified that an agent shall not return an SNMP error for an instrumented object, but shall return the identifies distinguished value.
4. Removed the IMPORT for **PrtInterpreterLangFamilyTC**, since the MIB doesn't acutally use this enum. In fact no enums used in the Attributes table actually need their enum TC imported into the Job Monitoring MIB, making the Job Monitoring MIB more extensible for adding new attributes that have textual conventions. The MIB now imports very little. Only **DateAndTime**, because it is used in the Queue table. Even the **TimeStamp** TC which is used in the attribute table, need not be imported into the **Job** Monitoring MIB.
5. Explained why there is both a jmJobState and a jmJobStateReasons object: so that the reasons can be extended without the monitoring application becoming confused as to what is happening, since the states won't be extended.
6. Clarified that **retained** is an optional state and its relationship to the **completed** state. Added conformance that only the **processing**, **needsAttention**, and **completed** states are required for conformance.
7. Changed the name of the **jmAttributeValueAsText** object to **jmAttributeValueAsOctets**, since the **DateAndTime** type is binary, not text. Changed the tag in the TC from "Text:" to "Octets".
8. Changed the name of the **mediaConsumed**(33) to **mediumConsumed**(33), since each entry is singular.

18.2 Changes to version 0.6, dated 1/23/97 to make version 0.7, dated 3/13/97

Changes to version 0.6, dated 1/23/97 to make version 0.7, dated 1/29/97:

1. Added PWG agreed boiler plate Status of this Memo.
2. Updated the Abstract from Ron's comments.
3. Incorporated Ron's re-written Introduction.
4. Explained the job set concept as representing a queue within a printer or a server, if the printer or server has several or the entire set of jobs, if the printer or server has only one queue.
5. Introduced the terminology of "attribute" instead of resource, since our table represents more than just resources now, as we agreed to move many non-resource

objects into it. Changed the name of the group and table from **jmResource** to **jmAttribute**.

6. Clarified that the **JmAttributesTypeTC** and **jmAttributesTable** contains information about the job, such as file name, document name, , as well as resources requested and/or consumed. Re-organized the attributes into groups of similar attributes.
7. Added more explanation about configuration 1 and 2 and added Configuration 3 as agreed to cover the case of a monitoring application that monitors a server not using SNMP while also monitoring using our MIB the printer(s) that the server controls.
8. Added more explanation of the security, internationalization, and IANA considerations.
9. Deleted the Job Set Group, since the monitoring application can find all the job sets via a Get.
10. Removed the **jmResourceUnits** object and specified the units in each **jmAttributeTypeIndex** enum. This makes it clearer what the units are and reduces the variability between agent implementations, thus making monitoring applications easier. Also cleanup the attribute names by adding the data type to the attribute name for those attributes that have more than one type that differs in the units (**Index** vs **Name**, **Name** vs. **Enum**, **DateAndTime** vs **TimeStamp**).
11. Added the **TimeStamp** data type as an alternative to **DateAndTime** and doubled the number of attributes that have to do with time.
12. Deleted the **JmQueuingAlgorithmTC** and **RmResourceUnitsTC** textual-conventions.
13. Added **other**(1) and **unknown**(2) to the **JmJobTypesTC** and moved the rest of the bits over.
14. Added **other**(1) to the **JmJobStatesTC.**
15. Added **jobPrinting**(45) to the **JmJobStateReasonsTC** to align with IPP.
16. Move 9 objects from the **jmJobTable** to the **JmAttributeTypeTC** and **jmAttributeTable,** making them attributes: **jobAccountName**, **jobComment**, **jobSourceChannelIndex**, **physicalDeviceName**, **jobTotalKOctets**, **jobKOctetsCompleted**, **jobSubmissionDateAndTime**, **jobSubmissionTimeStamp**, **jobStartedProcessingDateAndTime, jobStartedProcessingTimeStamp**, **jobCompletionDateAndTime**, **jobCompletionTimeStamp**. NOTE that some objects became two attributes as we have two forms of time. Also made the end of each name indicate the data type.
17. Added **Requested**, **Completed**, and **CompletedCurrentCopy** forms for impressions, sheets, and pages attributes.
18. Added: **other**(1), **outputBin**(9) attributes.
19. Added "CPU" to **processingCPUTime** attribute.

20. Added jmGeneralJobSetName so that the user could associate a name with a job set when the implementation had more than one job set. The name would typically be the queue name in such a case.

21. Added **jmGeneralNumberOfJobsCompleted** and renamed jmGeneralCurrentN**u**mberOfJobs to **jmGeneralNumberOfJobsToComplete**, so that a monitoring application can find out how many jobs have completed for the **jmCompletedTable** and how many are still to be comppleted. Their sum in the total number of jobs in the **jmJobTable**.

22. Clarified that **jmQueueIndex** shall be monitonically increasing which can change as new job arrive or the configuration changes.

23. Added the word **Queue** to make **jmQueueJobIndex** in the Queue table.

24. Clarifed that the **jmQueueJobIndex** and **jmJobIndex** shall not be 0 as required by SNMP for indexes. This gives agents that want to use the job-identifier that is generated by the system as the value for the **jmJobIndex** and **jmQueueJobIndex** a problem, if 0 is a legal value, such as in LPD.

25. Clarified the distinction betwen **jmJobName** and **jmJobComment** (now jobComment attribute): jmJobName is more of a name for identificaion purposes while jobComment is free form text that often isn't present and is intended to convey anything the submitting user wanted to convey usually to him/herself.

26. Clarified that -2 (unknown) shall be returned if the value of jmJobIndexNumber is unknown as in the Printer MIB convention.

27. Added "**OrQueue**" to make **jmJobDeviceNameOrQueueRequested**, since some didn't know which object to use for a system in which the user specifies a queue.

28. Added upper bound in **jmJobIndex** so that the MIB would compile.

29. Added "**Index**" to make **jmAttributeTypeIndex** object, since this object is both a type and an index.

30. Changed the name of the **jmResourceIndex** to **jmAttributeInstanceIndex**, since this index can be used for attributes that can have more than one instance per job, such as **fileName**, **documentFormat**, **outputBin**, etc.

31. Clarified that the jmAttributeInstanceIndex shall be the document number for those attributes that are one to one with a document, such as **fileName**(3) and **documentName**(4).

32. Replaced the **jmResourceAmount** with **jmAttributeValueAsInteger** and **jmAttributeValueAsText**

19. INDEX

This index includes the textual conventions, the objects, and the attributes. Textual conventions all start with the prefix: "**JM**" and end with the suffix: "**TC**". Objects all starts with the prefix: "**jm**" followed by the group name. Attributes are identified with enums, and so start with any lower case letter and have not special prefix.