INTERNET-DRAFT R. Bergman
Dataproducts Corp.
T. Hastings (editor)
Xerox Corporation
S. Isaacson
Novell, Inc.
H. Lewis
IBM Corp.
February 20, 1999

Job Monitoring MIB - V2.0
<draft-ietf-printmib-job-monitor-v2-00.txt>

Status of this Memo

Abstract

This document has been developed and approved by the Printer Working Group (PWG) as a PWG standard. It is intended to be distributed as an Informational RFC. This document provides a printer industry standard SNMP MIB 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 (1) in a printer or (2) in 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

Job Monitoring MIB

1 Introduction

This specification defines an official Printer Working Group (PWG) [PWG] standard SNMP MIB for the monitoring of jobs on network printers. This specification is being published as an IETF Information Document for the convenience of the Internet community. In consultation with the IETF Application Area Directors, it was concluded that this MIB specification properly belongs as an Information document, because this MIB monitors a service node on the network, rather than a network node proper.

The Job Monitoring MIB is intended to be implemented 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[iso-dpa], those protocols would be used to monitor and manage print jobs rather than using the Job Monitoring MIB.

The Job Monitoring MIB consists of a General Group, a Job Submission ID Group, a Job Group, and an Attribute Group. Each group is a table. All accessible objects are read-only. The General Group contains general information that applies to all jobs in a job set. The Job Submission ID table maps the job submission ID that the client uses to identify a job to the jmJobIndex that the Job Monitoring Agent uses to identify jobs in the Job and Attribute tables. The Job table contains the MANDATORY integer job state and status objects. The Attribute table consists of multiple entries per job that specify (1) job and document identification and parameters, (2) requested resources, and (3) consumed resources during and after job processing/printing. A larger number of job attributes are defined as textual conventions that an agent SHALL return if the server or device implements the functionality so represented and the agent has access to the information. The Attribute table provides access to job attributes by job index. An OPTIONAL Mirror Attribute table is defined which provides access to the same job attributes by attribute. A MANDATORY System Group provides a version number and objects that indicate which options and attributes are supported.

1.1 Types of Information in the MIB

The job MIB is intended to provide the following information for the indicated Role Models in the Printer MIB[print-mib] (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 user’s job (user queries).

Provide a timely indication 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 direct-connect (serial or parallel) and networked devices without unnecessary overhead or complexity, while also providing for higher end systems and devices.

1.2 Types of Job Monitoring Applications

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

1. Monitor a single job starting when the job is submitted and ending a defined period after the job completes. The Job Submission ID table provides the map to find the specific job to be monitored.
2. Monitor all ’active’ jobs in a queue, which this specification generalizes to a "job set". End users may use such a program when selecting a least busy printer, so the MIB is designed for such a program to start up quickly and find the information needed quickly without having to read all (completed) jobs in order to find the active jobs. System operators may also use such a program, in which case it would be running for a long period of time and may also be interested in the jobs that have completed. Finally such a program may be used to provide an enhanced console and logging capability.
3. Collect resource usage for accounting or system utilization purposes 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). Such a program is also expected to keep a shadow copy of the entire Job Attribute table including completed, canceled, and aborted jobs which the program updates on each polling cycle. Such a program polls at the rate of the persistence of the Attribute table. The design is not optimized to help such an application determine which jobs are completed, canceled, or aborted. Instead, the application SHOULD query each job that the application’s shadow copy shows was not complete, canceled, or aborted at the previous poll cycle to see if it is now complete or canceled, plus any new jobs that have been submitted.

The MIB provides a set of objects that represent a compatible subset of job and document attributes of the ISO DPA standard[iso-dpa] and the Internet Printing Protocol (IPP)[ipp-model], so that coherence is maintained between these two protocols and the information presented to end users and system operators by monitoring applications. 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)[tipsi], as well as with ones that do implement ISO DPA.

Thus the job monitoring MIB does not require implementation of either the ISO DPA or IPP protocols.

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 in alphabetical order.

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

Accounting Application: The SNMP management application that copies job information to some more permanent medium so that another application can perform accounting on the data for Accountants, Asset Managers, and Capacity Planners use.

Agent: The network entity that accepts SNMP requests from a *monitor* or *accounting application* and provides access to the instrumentation for managing jobs modeled by the management objects defined in the Job Monitoring MIB module for a *server* or a *device*.

Attribute: A name, value-pair that specifies a job or document instruction, a status, or a condition of a job or a document that has been submitted to a server or device. A particular 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, either because (1) the client supplied a value in the job submission protocol, (2) the document data contained an embedded attribute, or (3) the server or device supplied a default value. An agent MAY represent an attribute as an entry (row) in the Attribute table in this MIB in which entries are present only when necessary. Attributes are identified in this MIB by an enum.

Client: 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 over a serial or parallel port to a directly-connected device or over the network to a networked-connected device.

Device: A hardware entity that (1) interfaces to humans, such as a device that produces marks on paper or scans marks on paper to produce an electronic representation, (2) accesses digital media, such as CD-ROMs, or (3) interfaces electronically to another device, such as sends FAX data to another FAX device.

Document: A sub-section within a job that contains print data and *document instructions* that apply to just the document.

Document Instruction: 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.

End User: A user that uses a client to submit a print job. See "user".

Impression: For a print job, an impression is the passage of the entire side of a sheet by the marker, whether or not any marks are made and independent of the number of passes that the side makes past the marker. Thus a four pass color process counts as a single impression, as does highlight color. Impression counters count all kinds: monochrome, highlight color, and full process color, while full color counters only count full color impressions, and high light color counters only count high light color impressions.

One-sided processing involves one impression per sheet. Two-sided processing involves two impressions per sheet. If a two-sided document has an odd number of pages, the last sheet still counts as two impressions, if that sheet makes two passes through the marker or the marker marks on both sides of a sheet in a single pass. Two-up printing is the placement of two logical pages on one side of a sheet and so is still a single impression. See "page" and "sheet".

NOTE - Since impressions include blank sides, it is suggested that accounting application implementers consider charging for sheets, rather than impressions, possibly using the value of the sides attribute to select different charges for one-sided versus two-sided printing, since some users may think that impressions don’t include blank sides.

Internal Collation: The production of the sheets for each document copy performed within the printing device by making multiple passes over either the source or an intermediate representation of the document.

Job: A unit of work whose results are expected together without interjection of unrelated results. A job contains one or more *documents*.

Job Accounting: The activity of a management application of accessing the MIB and recording what happens to the job during and after the processing of the job.

Job Instruction: 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.

Job Monitoring (using SNMP): The activity of a management application of accessing the MIB and (1) identifying jobs in the job tables being processed by the server, printer or other devices, and (2) displaying information to the user about the processing of the job.

Job Monitoring Application: The SNMP management application that End Users, and System Operators 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. See "monitor".

Job Set: A group 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, so that the implementation only implements a single job set. 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.

Monitor: Short for Job Monitoring Application.

Page: A page is a logical division of the original source document. Number up is the imposition of more than one page on a single side of a sheet. See "impression" and "sheet" and "two-up".

Proxy: 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.

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

Printer: A *device* that puts marks on media.

Server: A network entity that accepts jobs from clients and in turn submits the jobs to *printers* and other *devices* that may be directly connected to the server via a serial or parallel port or may be on the network. A server MAY be a printer *supervisor* control program, or a print *spooler*.

Sheet: A sheet is a single instance of a medium, whether printing on one or both sides of the medium. See "impression" and "page".

SNMP Information Object: A name, value-pair that specifies an action, a status, or a condition in an SNMP MIB. Objects are identified in SNMP by an OBJECT IDENTIFIER.

Spooler: 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: 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.

Stacked: When a media sheet is placed in an output bin of a device.

Supervisor: 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.

System Operator: A user that uses a monitor to monitor the system and carries out tasks to keep the system running.

System Administrator: A user that specifies policy for the system.

Two-up: The placement of two pages on one side of a sheet so that each side or impressions counts as two pages. See "page" and "sheet".

User: A person that uses a client or a monitor. See "end user".

2.1 System Configurations for the Job Monitoring MIB

This section enumerates the three configurations in which the Job Monitoring MIB is intended to be used. To simplify the pictures, the *devices* are shown as *printers*. See section 1.1 entitled "Types of Information in the MIB".

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

2.1.1 Configuration 1 - client-printer

In the client-printer configuration 1, the client(s) submit jobs directly to the printer, either by some direct connect, or by network connection.

The job submitting client and/or monitoring application monitor jobs by communicating 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, plus a defined time period after the job enters the completed state in which accounting 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 2-1 - Configuration 1 - client-printer - agent in the printer

The Job Monitoring MIB is designed to support the following relationships (not shown in Figure 2-1):

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.

2.1.2 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. While configuration 2 is included, the design center for this MIB is configurations 1 and 3.

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

> 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. The Job Monitoring MIB agent obtains the required information from the printer by a method that is beyond the scope of this document. The agent in the server SHALL keep the job in the Job Monitoring MIB in the server as long as the job is in the printer, plus a defined time period after the job enters the completed state in which accounting programs can copy out the accounting data from the Job Monitoring MIB.

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

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

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

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

2.1.3 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. That server does *not* contain a Job Monitoring MIB agent.

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

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

2. A Job Monitoring MIB agent that is part of the printer to monitor jobs after the server passes the jobs to the printer.

        In such configurations, the server deletes its copy of the job from the server after submitting the job to the printer usually almost immediately (before the job does much processing, if any).

In configuration 3, 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====v==+
              #       |             |
              #       |    server   |
              #       |             |
              #       +----#-----+--+
              #    optional#     |
              #   ##########     |
              #   #              |
         +==+=v===v=+==+         |
         |  | agent |  |         |
         |  +-------+  |         |
         |   PRINTER   <---------+
         |             | Print Job Delivery Channel
         |             |
         +=============+
```

Figure 2-3 - 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 2-3):

     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 Managed Object Usage

This section describes the usage of the objects in the MIB.

3.1 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.

3.1.1 Conformance Terminology

This specification uses the verbs: "SHALL", "SHOULD", "MAY", and "NEED NOT" to specify conformance requirements according to RFC 2119 [RFC2119] 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.

3.1.2 Agent Conformance Requirements

A conforming agent:

1. SHALL implement *all* MANDATORY groups in this specification.
2. SHALL implement any attributes if (1) the server or device supports the functionality represented by the attribute and (2) the information is available to the agent.
3. SHOULD implement both forms of an attribute if it implements an attribute that permits a choice of INTEGER and OCTET STRING forms, since implementing both forms may help management applications by giving them a choice of representations, since the representation are equivalent. See the JmAttributeTypeTC textual-convention.

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

3.1.2.1 MIB II System Group objects

The Job Monitoring MIB agent SHALL implement all objects in the System Group of MIB-II[mib-II], whether the Printer MIB[print-mib] is implemented or not.

3.1.2.2 MIB II Interface Group objects

The Job Monitoring MIB agent SHALL implement all objects in the Interfaces Group of MIB-II[mib-II], whether the Printer MIB[print-mib] is implemented or not.

3.1.2.3 Printer MIB objects

If the agent is providing access to a device that is a printer, the agent SHALL implement all of the MANDATORY objects in the Printer MIB[print-mib] and all the objects in other MIBs that conformance to the Printer MIB requires, such as the Host Resources MIB[hr-mib]. If the agent is providing access to a server that controls one or more direct-connect or networked printers, the agent NEED NOT implement the Printer MIB and NEED NOT implement the Host Resources MIB.

3.1.3 Job Monitoring Application Conformance Requirements

A conforming job monitoring application:

1. SHALL accept the full syntactic range for all objects in all MANDATORY groups and all MANDATORY attributes that are required to be implemented by an agent according to Section 3.1.2 and SHALL either present them to the user or ignore them.
2. SHALL accept the full syntactic range for *all* attributes, including enum and bit values specified in this specification and additional ones that may be registered with the PWG and SHALL either present them to the user or ignore them. In particular, a conforming job monitoring application SHALL not malfunction when receiving any standard or registered enum or bit values. See Section 3.7 entitled "IANA and PWG Registration Considerations".
3. SHALL NOT fail when operating with agents that materialize attributes *after* the job has been submitted, as opposed to when the job is submitted.
4. SHALL, if it supports a time attribute, accept either form of the time attribute, since agents are free to implement either time form.

3.2 The Job Tables and the Oldest Active and Newest Active Indexes

The jmJobTable and jmAttributeTable contain objects and attributes, respectively, for each job in a job set. These first two indexes are:

1. jmGeneralJobSetIndex - which job set
2. jmJobIndex - which job in the job set

In order for a monitoring application to quickly find that active jobs (jobs in the pending, processing, or processingStopped states), the MIB contains two indexes:

1. jmGeneralOldestActiveJobIndex - the index of the active job that has been in the tables the longest.

2. jmGeneralNewestActiveJobIndex - the index of the active job that has been most recently added to the tables.

The agent SHALL assign the next incremental value of jmJobIndex to the job, when a new job is accepted by the server or device to which the agent is providing access. If the incremented value of jmJobIndex would exceed the implementation-defined maximum value for jmJobIndex, the agent SHALL 'wrap' back to 1. An agent uses the resulting value of jmJobIndex for storing information in the jmJobTable and the jmAttributeTable about the job.

It is recommended that the largest value for jmJobIndex be much larger than the maximum number of jobs that the implementation can contain at a single time, so as to minimize the premature re-use of a jmJobIndex value for a newer job while clients retain the same 'stale' value for an older job.

It is recommended that agents that are providing access to servers/devices that already allocate job-identifiers for jobs as integers use the same integer value for the jmJobIndex. Then management applications using this MIB and applications using other protocols will see the same job identifiers for the same jobs. Agents providing access to systems that contain jobs with a job identifier of 0 SHALL map the job identifier value 0 to a jmJobIndex value that is one higher than the highest job identifier value that any job can have on that system. Then only job 0 will have a different job-identifier value than the job's jmJobIndex value.

NOTE - If a server or device accepts jobs using multiple job submission protocols, it may be difficult for the agent to meet the recommendation to use the job-identifier values that the server or device assigns as the jmJobIndex value, unless the server/device assigns job-identifiers for each of its job submission protocols from the same job-identifier number space.

Each time a new job is accepted by the server or device that the agent is providing access to AND that job is to be ’active’ (pending, processing, or processingStopped, but not pendingHeld), the agent SHALL copy the value of the job’s jmJobIndex to the jmGeneralNewestActiveJobIndex object. If the new job is to be ’inactive’ (pendingHeld state), the agent SHALL not change the value of jmGeneralNewestActiveJobIndex object (though the agent SHALL assign the next incremental jmJobIndex value to the job).

When a job transitions from one of the ’active’ job states (pending, processing, processingStopped) to one of the ’inactive’ job states (pendingHeld, completed, canceled, or aborted), with a jmJobIndex value that matches the jmGeneralOldestActiveJobIndex object, the agent SHALL advance (or wrap) the value to the next oldest ’active’ job, if any. See the JmJobStateTC textual-convention for a definition of the job states.

Whenever a job transitions from one of the ’inactive’ job states to one of the ’active’ job states (from pendingHeld to pending or processing), the agent SHALL update the value of either the jmGeneralOldestActiveJobIndex or the jmGeneralNewestActiveJobIndex objects, or both, if the job’s jmJobIndex value is outside the range between jmGeneralOldestActiveJobIndex and jmGeneralNewestActiveJobIndex.

When all jobs become ’inactive’, i.e., enter the pendingHeld, completed, canceled, or aborted states, the agent SHALL set the value of both the jmGeneralOldestActiveJobIndex and jmGeneralNewestActiveJobIndex objects to 0.

NOTE - Applications that wish to efficiently access all of the active jobs MAY use jmGeneralOldestActiveJobIndex value to start with the oldest active job and continue until they reach the index value equal to jmGeneralNewestActiveJobIndex, skipping over any pendingHeld, completed, canceled, or aborted jobs that might intervene.

If an application detects that the jmGeneralNewestActiveJobIndex is smaller than jmGeneralOldestActiveJobIndex, the job index has wrapped. In this case, the application SHALL reset the index to 1 when the end of the table is reached and continue the GetNext operations to find the rest of the active jobs.

NOTE - Applications detect the end of the jmAttributeTable table when the OID returned by the GetNext operation is an OID in a different MIB. There is no object in this MIB that specifies the maximum value for the jmJobIndex supported by the implementation.

When the server or device is power-cycled, the agent SHALL remember the next jmJobIndex value to be assigned, so that new jobs are not assigned the same jmJobIndex as recent jobs before the power cycle.

3.3 The Attribute Mechanism and the Attribute Table(s)

Attributes are similar to information objects, except that attributes are identified by an enum, instead of an OID, so that attributes may be registered without requiring a new MIB. Also an implementation that does not have the functionality represented by the attribute can omit the attribute entirely, rather than having to return a distinguished value. The agent is free to materialize an attribute in the jmAttributeTable as soon as the agent is aware of the value of the attribute.

The agent materializes job attributes in a four-indexed jmAttributeTable:

1. jmGeneralJobSetIndex - which job set
2. jmJobIndex - which job in the job set
3. jmAttributeTypeIndex - which attribute
4. jmAttributeInstanceIndex - which attribute instance for those attributes that can have multiple values per job.

With this order of table indexing, an application can obtain all of the attributes of a particular job using SNMPv1 GetNext or SNMPv2 GetBulk.

An OPTIONAL mirror table, called jmMirrorAttrTable, provides access to the same job attributes, but with a different order to the indexes:

1. jmAttributeTypeIndex - which attribute
2. jmGeneralJobSetIndex - which job set
3. jmJobIndex - which job in the job set
4. jmAttributeInstanceIndex - which attribute instance for those attributes that can have multiple values per job.

With this order of table indexing, an application can obtain selected attributes of a number of jobs using SNMPv1 GetNext or SNMPv2 GetBulk. A management application can determine whether or not this table is implemented (even when the table is empty) by querying the jmSystemOptionSupport object.

Some attributes represent information about a job, such as a file-name, a document-name, a submission-time or a completion time. Other attributes represent resources required, e.g., a medium or a colorant, etc. to process the job before the job starts processing OR to indicate the amount of the resource consumed during and after processing, e.g., pages completed or impressions completed. If both a required and a consumed value of a resource is needed, this specification assigns two separate attribute enums in the textual convention.

NOTE - The table of contents lists all the attributes in order. This order is the order of enum assignments which is the order that the SNMP GetNext operation returns attributes. Most attributes apply to all three configurations covered by this MIB specification (see section 2.1 entitled "System Configurations for the Job Monitoring MIB"). Those attributes that apply to a particular configuration are indicated as 'Configuration *n*:' and SHALL NOT be used with other configurations.

3.3.1 Conformance of Attribute Implementation

An agent SHALL implement any attribute if (1) the server or device supports the functionality represented by the attribute and (2) the information is available to the agent. The agent MAY create the attribute row in the jmAttributeTable when the information is available or MAY create the row earlier with the designated 'unknown' value appropriate for that attribute. See next section.

If the server or device does not implement or does not provide access to the information about an attribute, the agent SHOULD NOT create the corresponding row in the jmAttributeTable.

3.3.2 Useful, 'Unknown', and 'Other' Values for Objects and Attributes

Some attributes have a 'useful' Integer32 value, some have a 'useful' OCTET STRING value, some MAY have either or both depending on implementation, and some MUST have both. See the JmAttributeTypeTC textual convention for the specification of each attribute.

SNMP requires that if an object cannot be implemented because its values cannot be accessed, then a compliant agent SHALL return an SNMP error in SNMPv1 or an exception value in SNMPv2. However, this MIB has been designed so that 'all' objects can and SHALL be implemented by an agent, so that neither the SNMPv1 error nor the SNMPv2 exception value SHALL be generated by the agent. This MIB has also been designed so that when an agent materializes an attribute, the agent SHALL materialize a row consisting of both the jmAttributeValueAsInteger and jmAttributeValueAsOctets objects.

In general, values for objects and attributes have been chosen so that a management application will be able to determine whether a 'useful', 'unknown', or 'other' value is available. When a useful value is not available for an object, that agent SHALL return a zero-length string for octet strings, the value 'unknown(2)' for enums, a '0' value for an object that represents an index in another table, and a value '-2' for counting integers.

Since each attribute is represented by a row consisting of both the jmAttributeValueAsInteger and jmAttributeValueAsOctets MANDATORY objects, SNMP requires that the agent SHALL always create an attribute row with both objects specified. However, for most attributes the agent SHALL return a "useful" value for one of the objects and SHALL return the 'other' value for the other object. For integer only attributes, the agent SHALL always return a zero-length string value for the jmAttributeValueAsOctets object. For octet string only attributes, the agent SHALL always return a '-1' value for the jmAttributeValueAsInteger object.

3.3.3 Index Value Attributes

A number of attributes are indexes in other tables. Such attribute names end with the word 'Index'. If the agent has not (yet) assigned an index value for a particular index attribute for a job, the agent SHALL either: (1) return the value 0 or (2) *not* add this attribute to the jmAttributeTable until the index value is assigned. In the interests of brevity, the semantics for 0 is specified once here and is *not* repeated for each index attribute specification and a DEFVAL of 0 is implied, even though the DEFVAL for jmAttributeValueAsInteger is -2.

3.3.4 Data Sub-types and Attribute Naming Conventions

Many attributes are sub-typed to give a more specific data type than Integer32 or OCTET STRING. The data sub-type of each attribute is indicated on the first line(s) of the description. Some attributes have several different data sub-type representations. When an attribute has both an Integer32 data sub-type and an OCTET STRING data sub-type, the attribute can be represented in a single row in the jmAttributeTable. In this case, the data sub-type name is not included as the last part of the name of the attribute, e.g., documentFormat(38) which is both an enum and/or a name. When the data sub-types cannot be represented by a single row in the jmAttributeTable, each such representation is considered a separate attribute and is assigned a separate name and enum value. For these attributes, the name of the data sub-type is the last part of the name of the attribute: Name, Index, DateAndTime, TimeStamp, etc. For example, documentFormatIndex(37) is an index.

NOTE: The Table of Contents also lists the data sub-type and/or data sub-types of each attribute, using the textual-convention name when such is defined. The following abbreviations are used in the Table of Contents as shown:

'Int32(-2..)'	Integer32 (-2..2147483647)
'Int32(0..)'	Integer32 (0..2147483647)
'Int32(1..)'	Integer32 (1..2147483647)
'Int32(m..n)'	For all other Integer ranges, the lower and upper bound of the range is indicated.
'UTF8String63'	JmUTF8StringTC (SIZE(0..63))
'JobString63'	JmJobStringTC (SIZE(0..63))
'Octets63'	OCTET STRING (SIZE(0..63))
'Octets(m..n)'	For all other OCTET STRING ranges, the exact range is indicated.

3.3.5 Single-Value (Row) Versus Multi-Value (MULTI-ROW) Attributes

Most attributes have only one row per job. However, a few attributes can have multiple values per job or even per document, where each value is a separate row in the jmAttributeTable. Unless indicated with 'MULTI-ROW:' in the JmAttributeTypeTC description, an agent SHALL ensure that each attribute occurs only once in the jmAttributeTable for a job. Most of the 'MULTI-ROW' attributes do not allow duplicate values, i.e., the agent SHALL ensure that each value occurs only once for a job. Only if the specification of the 'MULTI-ROW' attribute also says "There is no restriction on the same xxx occurring in multiple rows" can the agent allow duplicate values to occur for the job.

NOTE - Duplicates are allowed for 'extensive' 'MULTI-ROW' attributes, such as fileName(34) or documentName(35) which are specified to be 'per-document' attributes, but are *not* allowed for 'intensive' 'MULTI-ROW' attributes, such as mediumConsumed(171) and documentFormat(38) which are specified to be 'per-job' attributes.

3.3.6 Requested Objects and Attributes

A number of objects and attributes record requirements for the job. Such object and attribute names end with the word 'Requested'. In the interests of brevity, the phrase 'requested' means: (1) requested by the client (or intervening server) in the job submission protocol and may also mean (2) embedded in the submitted document data, and/or (3) defaulted by the recipient device or server with the same semantics as if the requester had supplied, depending on implementation. Also if a value is supplied by the job submission client, and the server/device determines a better value, through processing or other means, the agent MAY return that better value for such object and attribute.

3.3.7 Consumption Attributes

A number of objects and attributes record consumption. Such attribute names end with the word 'Completed' or 'Consumed'. If the job has not yet consumed what that resource is metering, the agent either: (1) SHALL return the value 0 or (2) SHALL *not* add this attribute to the jmAttributeTable until the consumption begins. In the interests of brevity, the semantics for 0 is specified once here and is *not* repeated for each consumption attribute specification and a DEFVAL of 0 is implied, even though the DEFVAL for jmAttributeValueAsInteger is -2.

3.3.8 Attribute Specifications

This section specifies the job attributes.

In the following definitions of the attributes, each description indicates whether the useful value of the attribute SHALL be represented using the jmAttributeValueAsInteger or the jmAttributeValueAsOctets objects by the initial tag: 'INTEGER:' or 'OCTETS:', respectively.

Some attributes allow the agent implementer a choice of useful values of either an integer, an octet string representation, or both, depending on implementation. These attributes are indicated with 'INTEGER:' AND/OR 'OCTETS:' tags.

A very few attributes require both objects at the same time to represent a pair of useful values (see mediumConsumed(171)). These attributes are indicated with 'INTEGER:' AND 'OCTETS:' tags. See the jmAttributeGroup for the descriptions of these two MANDATORY objects.

A management application can determine which attributes are supported and whether the integer and/or the octet string values are supported with meaningful value by querying the jmSystemAttrIntegerSupport and jmSystemAttrOctetsSupport objects, respectively. Management applications can also determine which supported attributes might support more than one integer value or more than one octet string value by querying jmSystemAttrMultiRowSupport.

These support bits are indicated in hex for each range in the line starting with "support bits starting:". Note: these objects permit a management application to determine the degree of support, even when there are no jobs present in the system. They also permit management middleware to fetch all attribute values for all jobs, including future extensions, and keep them updated for one or more management applications at the same time.

NOTE - The enum assignments are grouped logically with values assigned in groups of 20, so that additional values may be registered in the future and assigned a value that is part of their logical grouping.

Values in the range 2**30 to 2**31-1 are reserved for private or experimental usage. This range corresponds to the same range reserved in IPP. Implementers are warned that use of such values may conflict with other implementations. Implementers are encouraged to request registration of enum values following the procedures in Section 3.7.1.

NOTE: No attribute name exceeds 31 characters.

The standard attribute types are:

```
        jmAttributeTypeIndex              Datatype
        --------------------              --------

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

        +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
        + Job State attributes (3 - 19 decimal)
        +
        + The following attributes specify the state of a job.
        +       support bits starting: { ’10’H }
        +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

        jobStateReasons2(3),              JmJobStateReasons2TC
            INTEGER:  Additional information about the job’s current
            state that augments the jmJobState object.  See the
            description under the JmJobStateReasons1TC textual-
            convention.

        jobStateReasons3(4),              JmJobStateReasons3TC
            INTEGER:  Additional information about the job’s current
            state that augments the jmJobState object.  See the
            description under JmJobStateReasons1TC textual-convention.

        jobStateReasons4(5),              JmJobStateReasons4TC
            INTEGER:  Additional information about the job’s current
            state that augments the jmJobState object.  See the
            description under JmJobStateReasons1TC textual-convention.
        processingMessage(6),             JmUTF8StringTC (SIZE(0..63))
            OCTETS:  MULTI-ROW:  A coded character set message that is
            generated by the server or device during the processing of
            the job as a simple form of processing log to show progress
            and any problems.  The natural language of each value is
            specified by the corresponding
            processingMessageNaturalLangTag(7) value.

            NOTE - This attribute is intended for such conditions as
            interpreter messages, rather than being the printable form
            of the jmJobState and jmJobStateReasons1 objects and
            jobStateReasons2, jobStateReasons3, and jobStateReasons4
            attributes.  In order to produce a localized printable form
            of these job state objects/attribute, a management
            application SHOULD produce a message from their enum and
            bit values.

            NOTE - There is no job description attribute in IPP/1.0
            that corresponds to this attribute and this attribute does
            not correspond to the IPP/1.0 'job-state-message' job
            description attribute, which is just a printable form of
            the IPP 'job-state' and 'job-state-reasons' job attributes.

            There is no restriction for the same message occurring in
            multiple rows.

        processingMessageNaturalLangTag(7),   OCTET STRING(SIZE(0..63))
            OCTETS:  MULTI-ROW:  The natural language of the
            corresponding processingMessage(6) attribute value.  See
            section 3.6.1, entitled 'Text generated by the server or
            device'.

            If the agent does not know the natural language of the job
            processing message, the agent SHALL either (1) return a
            zero length string value for the
            processingMessageNaturalLangTag(7) attribute or (2) not
            return the processingMessageNaturalLangTag(7) attribute for
            the job.

            There is no restriction for the same tag occurring in
            multiple rows, since when this attribute is implemented, it
            SHOULD have a value row for each corresponding
            processingMessage(6) attribute value row.
```

jobCodedCharSet(8), CodedCharSet

INTEGER: The MIBenum identifier of the coded character set that the agent is using to represent coded character set objects and attributes of type 'JmJobStringTC'. These coded character set objects and attributes are either: (1) supplied by the job submitting client or (2) defaulted by the server or device when omitted by the job submitting client. The agent SHALL represent these objects and attributes in the MIB either (1) in the coded character set as they were submitted or (2) MAY convert the coded character set to another coded character set or encoding scheme as identified by the jobCodedCharSet(8) attribute. See section 3.6.2, entitled 'Text supplied by the job submitter'.

These MIBenum values are assigned by IANA [IANA-charsets] when the coded character sets are registered. The coded character set SHALL be one of the ones registered with IANA [IANA] and the enum value uses the CodedCharSet textual-convention from the Printer MIB. See the JmJobStringTC textual-convention.

If the agent does not know what coded character set was used by the job submitting client, the agent SHALL either (1) return the 'unknown(2)' value for the jobCodedCharSet(8) attribute or (2) not return the jobCodedCharSet(8) attribute for the job.

jobNaturalLanguageTag(9), OCTET STRING(SIZE(0..63))

OCTETS: The natural language of the job attributes supplied by the job submitter or defaulted by the server or device for the job, i.e., all objects and attributes represented by the 'JmJobStringTC' textual-convention, such as jobName, mediumRequested, etc. See Section 3.6.2, entitled 'Text supplied by the job submitter'.

If the agent does not know what natural language was used by the job submitting client, the agent SHALL either (1) return a zero length string value for the jobNaturalLanguageTag(9) attribute or (2) not return jobNaturalLanguageTag(9) attribute for the job.

```
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+ Job Identification attributes (20 - 49 decimal)
+
+ The following attributes help an end user, a system
+ operator, or an accounting program identify a job.
+       support bits starting: { '000008'H }
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

jobURI(20),                        OCTET STRING(SIZE(0..63))
    OCTETS:  MULTI-ROW:  The job's Universal Resource
    Identifier (URI) [RFC1738].  See IPP [ipp-model] for
    example usage.

    NOTE - The agent may be able to generate this value on each
    SNMP Get operation from smaller values, rather than having
    to store the entire URI.

    If the URI exceeds 63 octets, the agent SHALL use multiple
    values, with the next 63 octets coming in the second value,
    etc.

    NOTE - IPP [ipp-model] has a 1023-octet maximum length for
    a URI, though the URI standard itself and HTTP/1.1 specify
    no maximum length.

jobAccountName(21),                OCTET STRING(SIZE(0..63))
    OCTETS:  Arbitrary binary information which MAY be coded
    character set data or encrypted data supplied by the
    submitting user for use by accounting services to allocate
    or categorize charges for services provided, such as a
    customer account name or number.

    NOTE: This attribute NEED NOT be printable characters.

serverAssignedJobName(22),        JmJobStringTC (SIZE(0..63))
    OCTETS:  Configuration 3 only:  The human readable string
    name, number, or ID of the job as assigned by the server
    that submitted the job to the device that the agent is
    providing access to with this MIB.

    NOTE - This attribute is intended for enabling a user to
    find his/her job that a server submitted to a device when
    either the client does not support the jmJobSubmissionID or
    the server does not pass the jmJobSubmissionID through to
    the device.
```

```
          jobName(23),                        JmJobStringTC (SIZE(0..63))
              OCTETS:  The human readable string name of the job as
              assigned by the submitting user to help the user
              distinguish between his/her various jobs.  This name does
              not need to be unique.

              This attribute is intended for enabling a user or the
              user's application to convey a job name that MAY be printed
              on a start sheet, returned in a query result, or used in
              notification or logging messages.

              In order to assist users to find their jobs for job
              submission protocols that don't supply a jmJobSubmissionID,
              the agent SHOULD maintain the jobName attribute for the
              time specified by the jmGeneralJobPersistence object,
              rather than the (shorter) jmGeneralAttributePersistence
              object.

              If this attribute is not specified when the job is
              submitted, no job name is assumed, but implementation
              specific defaults are allowed, such as the value of the
              documentName attribute of the first document in the job or
              the fileName attribute of the first document in the job.

              The jobName attribute is distinguished from the jobComment
              attribute, in that the jobName attribute is intended to
              permit the submitting user to distinguish between different
              jobs that he/she has submitted.  The jobComment attribute
              is intended to be free form additional information that a
              user might wish to use to communicate with himself/herself,
              such as a reminder of what to do with the results or to
              indicate a different set of input parameters were tried in
              several different job submissions.
```

jobServiceTypes(24),   JmJobServiceTypesTC

INTEGER: Specifies the type(s) of service to which the job has been submitted (print, fax, scan, etc.). The service type is bit encoded with each job service type so that more general and arbitrary services can be created, such as services with more than one destination type, or ones with only a source or only a destination. For example, a job service might scan, faxOut, and print a single job. In this case, three bits would be set in the jobServiceTypes attribute, corresponding to the hexadecimal values: 0x8 + 0x20 + 0x4, respectively, yielding: 0x2C.

Whether this attribute is set from a job attribute supplied by the job submission client or is set by the recipient job submission server or device depends on the job submission protocol. This attribute SHALL be implemented if the server or device has other types in addition to or instead of printing.

One of the purposes of this attribute is to permit a requester to filter out jobs that are not of interest. For example, a printer operator may only be interested in jobs that include printing.

jobSourceChannelIndex(25),   Integer32 (0..2147483647)

INTEGER: The index of the row in the associated Printer MIB[print-mib] of the channel which is the source of the print job.

jobSourcePlatformType(26),   JmJobSourcePlatformTypeTC

INTEGER: The source platform type of the immediate upstream submitter that submitted the job to the server (configuration 2) or device (configuration 1 and 3) to which the agent is providing access. For configuration 1, this is the type of the client that submitted the job to the device; for configuration 2, this is the type of the client that submitted the job to the server; and for configuration 3, this is the type of the server that submitted the job to the device.

submittingServerName(27),   JmJobStringTC (SIZE(0..63))

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

submittingApplicationName(28),   JmJobStringTC (SIZE(0..63))

OCTETS: The name of the client application (not the server in configuration 3) that submitted the job to the server or device.

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

        deviceNameRequested(30),          JmJobStringTC (SIZE(0..63))
            OCTETS:  The administratively defined coded character set
            name of the target device requested by the submitting user.
            For configuration 1, its value corresponds to the Printer
            MIB[print-mib]: prtGeneralPrinterName object.  For
            configuration 2 and 3, its value is the name of the logical
            or physical device that the user supplied to indicate to
            the server on which device(s) they wanted the job to be
            processed.

        queueNameRequested(31),           JmJobStringTC (SIZE(0..63))
            OCTETS:  The administratively defined coded character set
            name of the target queue requested by the submitting user.
            For configuration 1, its value corresponds to the queue in
            the device for which the agent is providing access.  For
            configuration 2 and 3, its value is the name of the queue
            that the user supplied to indicate to the server on which
            device(s) they wanted the job to be processed.

            NOTE - typically an implementation SHOULD support either
            the deviceNameRequested or queueNameRequested attribute,
            but not both.

        physicalDevice(32),               hrDeviceIndex
                                          AND/OR
                                          JmUTF8StringTC (SIZE(0..63))
            INTEGER:  MULTI-ROW:  The index of the physical device MIB
            instance requested/used, such as the Printer MIB[print-
            mib].  This value is an hrDeviceIndex value.  See the Host
            Resources MIB[hr-mib].

            AND/OR

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

        numberOfDocuments(33),            Integer32 (-2..2147483647)
            INTEGER:  The number of documents in this job.

            The agent SHOULD return this attribute if the job has more
            than one document.
        fileName(34),                     JmJobStringTC (SIZE(0..63))
            OCTETS:  MULTI-ROW:  The coded character set file name or
            URI[URI-spec] of the document.

            There is no restriction on the same file name occurring in
            multiple rows.

        documentName(35),                 JmJobStringTC (SIZE(0..63))
            OCTETS:  MULTI-ROW:  The coded character set name of the
            document.

            There is no restriction on the same document name occurring
            in multiple rows.

        jobComment(36),                   JmJobStringTC (SIZE(0..63))
            OCTETS:  An arbitrary human-readable coded character text
            string supplied by the submitting user or the job
            submitting application program for any purpose.  For
            example, a user might indicate what he/she is going to do
            with the printed output or the job submitting application
            program might indicate how the document was produced.

            The jobComment attribute is not intended to be a name; see
            the jobName attribute.

        documentFormatIndex(37),          Integer32 (0..2147483647)
            INTEGER:  MULTI-ROW:  The index in the prtInterpreterTable
            in the Printer MIB[print-mib] of the page description
            language (PDL) or control language interpreter that this
            job requires/uses.  A document or a job MAY use more than
            one PDL or control language.

            NOTE - As with all intensive attributes where multiple rows
            are allowed, there SHALL be only one distinct row for each
            distinct interpreter; 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 an
            agent SHALL use the documentFormat attribute instead.
        documentFormat(38),               PrtInterpreterLangFamilyTC
                                          AND/OR
                                          OCTET STRING(SIZE(0..63))
            INTEGER:  MULTI-ROW:  The interpreter language family
            corresponding to the Printer MIB[print-mib]
            prtInterpreterLangFamily object, that this job
            requires/uses.  A document or a job MAY use more than one
            PDL or control language.

            AND/OR

            OCTETS:  MULTI-ROW:  The document format registered as a
            media type[iana-media-types], i.e., the name of the MIME
            content-type/subtype.  Examples: 'application/postscript',
            'application/vnd.hp-PCL', 'application/pdf', 'text/plain'
            (US-ASCII SHALL be assumed), 'text/plain; charset=iso-8859-
            1', and 'application/octet-stream'.  The IPP 'document-
            format' job attribute uses these same values with the same
            semantics.  See the IPP [ipp-model] 'mimeMediaType'
            attribute syntax and the document-format attribute for
            further examples and explanation.

        +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
        + Job Parameter attributes (50 - 67 decimal)
        +
        + The following attributes represent input parameters
        + supplied by the submitting client in the job submission
        + protocol.
        +       support bits starting: { '00000000 000020'H }
        +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

        jobPriority(50),                  Integer32 (-2..100)
            INTEGER:  The priority for scheduling the job.  It is used
            by servers and devices that employ a priority-based
            scheduling algorithm.

            A higher value specifies a higher priority.  The value 1 is
            defined to indicate the lowest possible priority (a job
            which a priority-based scheduling algorithm SHALL pass over
            in favor of higher priority jobs).  The value 100 is
            defined to indicate the highest possible priority.
            Priority is expected to be evenly or 'normally' distributed
            across this range.  The mapping of vendor-defined priority
            over this range is implementation-specific.  -2 indicates
            unknown.
```

jobProcessAfterDateAndTime(51), DateAndTime (SNMPv2-TC)

OCTETS: The calendar date and time of day after which the job SHALL become a candidate to be scheduled for processing. If the value of this attribute is in the future, the server SHALL set the value of the job’s jmJobState object to pendingHeld and add the jobProcessAfterSpecified bit value to the job’s jmJobStateReasons1 object. When the specified date and time arrives, the server SHALL remove the jobProcessAfterSpecified bit value from the job’s jmJobStateReasons1 object and, if no other reasons remain, SHALL change the job’s jmJobState object to pending.

jobHold(52), JmBooleanTC

INTEGER: If the value is ’true(4)’, a client has explicitly specified that the job is to be held until explicitly released. Until the job is explicitly released by a client, the job SHALL be in the pendingHeld state with the jobHoldSpecified value in the jmJobStateReasons1 attribute.

jobHoldUntil(53), JmJobStringTC (SIZE(0..63))

OCTETS: The named time period during which the job SHALL become a candidate for processing, such as ’evening’, ’night’, ’weekend’, ’second-shift’, ’third-shift’, etc., (supported values configured by the system administrator). See IPP [ipp-model] for the standard keyword values. Until that time period arrives, the job SHALL be in the pendingHeld state with the jobHoldUntilSpecified value in the jmJobStateReasons1 object. The value ’no-hold’ SHALL indicate explicitly that no time period has been specified; the absence of this attribute SHALL indicate implicitly that no time period has been specified.

outputBin(54), Integer32 (0..2147483647)
AND/OR
JmJobStringTC (SIZE(0..63))

INTEGER: MULTI-ROW: The output subunit index in the Printer MIB[print-mib]

AND/OR

OCTETS: MULTI-ROW: the name or number (represented as ASCII digits) of the output bin to which all or part of the job is placed in.

sides(55), Integer32 (-2..2)

INTEGER: MULTI-ROW: The number of sides, ’1’ or ’2’, that any document in this job requires/used.

```
        finishing(56),                    JmFinishingTC
            INTEGER:  MULTI-ROW:  Type of finishing that any document
            in this job requires/used.


        +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
        + Image Quality attributes (requested and consumed) (70 - 87)
        +
        + For devices that can vary the image quality.
        +       support bits starting: { '00000000 00000000 02'H }
        +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

        printQualityRequested(70),        JmPrintQualityTC
            INTEGER:  MULTI-ROW:  The print quality selection requested
            for a document in the job for printers that allow quality
            differentiation.

        printQualityUsed(71),             JmPrintQualityTC
            INTEGER:  MULTI-ROW:  The print quality selection actually
            used by a document in the job for printers that allow
            quality differentiation.

        printerResolutionRequested(72),   JmPrinterResolutionTC
            OCTETS:  MULTI-ROW:  The printer resolution requested for a
            document in the job for printers that support resolution
            selection.

        printerResolutionUsed(73),        JmPrinterResolutionTC
            OCTETS:  MULTI-ROW:  The printer resolution actually used
            by a document in the job for printers that support
            resolution selection.

        tonerEcomonyRequested(74),        JmTonerEconomyTC
            INTEGER:  MULTI-ROW:  The toner economy selection requested
            for documents in the job for printers that allow toner
            economy differentiation.

        tonerEcomonyUsed(75),             JmTonerEconomyTC
            INTEGER:  MULTI-ROW:  The toner economy selection actually
            used by documents in the job for printers that allow toner
            economy differentiation.

        tonerDensityRequested(76)         Integer32 (-2..100)
            INTEGER:  MULTI-ROW:  The toner density requested for a
            document in this job for devices that can vary toner
            density levels.  Level 1 is the lowest density and level
            100 is the highest density level.  Devices with a smaller
            range, SHALL map the 1-100 range evenly onto the
            implemented range.
```

```
tonerDensityUsed(77),            Integer32 (-2..100)
    INTEGER:  MULTI-ROW:  The toner density used by documents
    in this job for devices that can vary toner density levels.
    Level 1 is the lowest density and level 100 is the highest
    density level.  Devices with a smaller range, SHALL map the
    1-100 range evenly onto the implemented range.

+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+ Job Progress attributes (requested and consumed) (90-109)
+
+ Pairs of these attributes can be used by monitoring
+ applications to show an indication of relative progress
+ to users.  See section 3.4, entitled:
+ 'Monitoring Job Progress'.
+    support bits starting: { '00000000 00000000 00000020'H }
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

jobCopiesRequested(90),          Integer32 (-2..2147483647)
    INTEGER:  The number of copies of the entire job that are
    to be produced.

jobCopiesCompleted(91),          Integer32 (-2..2147483647)
    INTEGER:  The number of copies of the entire job that have
    been completed so far.

documentCopiesRequested(92),     Integer32 (-2..2147483647)
    INTEGER:  The total count of the number of document copies
    requested for the job as a whole.  If there are documents
    A, B, and C, and document B is specified to produce 4
    copies, the number of document copies requested is 6 for
    the job.

    This attribute SHALL be used only when a job has multiple
    documents.  The jobCopiesRequested attribute SHALL be used
    when the job has only one document.

documentCopiesCompleted(93),     Integer32 (-2..2147483647)
    INTEGER:  The total count of the number of document copies
    completed so far for the job as a whole.  If there are
    documents A, B, and C, and document B is specified to
    produce 4 copies, the number of document copies starts a 0
    and runs up to 6 for the job as the job processes.

    This attribute SHALL be used only when a job has multiple
    documents.  The jobCopiesCompleted attribute SHALL be used
    when the job has only one document.
```

```
        jobKOctetsTransferred(94),        Integer32 (-2..2147483647)
            INTEGER:  The number of K (1024) octets transferred to the
            server or device to which the agent is providing access.
            This count is independent of the number of copies of the
            job or documents that will be produced, but it is only a
            measure of the number of bytes transferred to the server or
            device.

            The agent SHALL round the actual number of octets
            transferred up to the next higher K.  Thus 0 octets SHALL
            be represented as '0', 1-1024 octets SHALL BE represented
            as '1', 1025-2048 SHALL be '2', etc.  When the job
            completes, the values of the jmJobKOctetsPerCopyRequested
            object and the jobKOctetsTransferred attribute SHALL be
            equal.

            NOTE - The jobKOctetsTransferred can be used with the
            jmJobKOctetsPerCopyRequested object in order to produce a
            relative indication of the progress of the job for agents
            that do not implement the jmJobKOctetsProcessed object.

        sheetCompletedCopyNumber(95),     Integer32 (-2..2147483647)
            INTEGER:  The number of the copy being stacked for the
            current document.  This number starts at 0, is set to 1
            when the first sheet of the first copy for each document is
            being stacked and is equal to n where n is the nth sheet
            stacked in the current document copy.  See section 3.4 ,
            entitled 'Monitoring Job Progress'.

        sheetCompletedDocumentNumber(96), Integer32 (-2..2147483647)
            INTEGER:  The ordinal number of the document in the job
            that is currently being stacked.  This number starts at 0,
            increments to 1 when the first sheet of the first document
            in the job is being stacked, and is equal to n where n is
            the nth document in the job, starting with 1.

            Implementations that only support one document jobs SHOULD
            NOT implement this attribute.

        jobCollationType(97),             JmJobCollationTypeTC
            INTEGER:  The type of job collation. See also Section 3.4,
            entitled 'Monitoring Job Progress'.
        +++++++++++++++++++++++++++++++++++++++++++++++++++++++++
        + Impression attributes (110 - 129 decimal)
        +
        + See the definition of the terms 'impression', 'sheet',
        + and 'page' in Section 2.
        +
        + See also jmJobImpressionsPerCopyRequested and
        + jmJobImpressionsCompleted objects in the jmJobTable.
        + support bits starting: { '00000000 00000000 00000000 0002'H }
        +++++++++++++++++++++++++++++++++++++++++++++++++++++++++

        impressionsSpooled(110),          Integer32 (-2..2147483647)
            INTEGER:  The number of impressions spooled to the server
            or device for the job so far.

        impressionsSentToDevice(111),     Integer32 (-2..2147483647)
            INTEGER:  The number of impressions sent to the device for
            the job so far.

        impressionsInterpreted(112),      Integer32 (-2..2147483647)
            INTEGER:  The number of impressions interpreted for the job
            so far.

        impressionsCompletedCurrentCopy(113),
                                          Integer32 (-2..2147483647)
            INTEGER:  The number of impressions completed by the device
            for the current copy of the current document so far.  For
            printing, the impressions completed includes interpreting,
            marking, and stacking the output.  For other types of job
            services, the number of impressions completed includes the
            number of impressions processed.

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

        fullColorImpressionsCompleted(114), Integer32 (-2..2147483647)
            INTEGER:  The number of full color impressions completed by
            the device for this job so far.  For printing, the
            impressions completed includes interpreting, marking, and
            stacking the output.  For other types of job services, the
            number of impressions completed includes the number of
            impressions processed. Full color impressions are typically
            defined as those requiring 3 or more colorants, but this
            MAY vary by implementation.  In any case, the value of this
            attribute counts by 1 for each side that has full color,
            not by the number of colors per side (and the other
            impression counters are incremented, except
            highlightColorImpressionsCompleted(115)).
        highlightColorImpressionsCompleted(115),
                                          Integer32 (-2..2147483647)
            INTEGER:  The number of highlight color impressions
            completed by the device for this job so far.  For printing,
            the impressions completed includes interpreting, marking,
            and stacking the output.  For other types of job services,
            the number of impressions completed includes the number of
            impressions processed.  Highlight color impressions are
            typically defined as those requiring black plus one other
            colorant, but this MAY vary by implementation.  In any
            case, the value of this attribute counts by 1 for each side
            that has highlight color (and the other impression counters
            are incremented, except
            fullColorImpressionsCompleted(114)).

        +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
        + Page attributes (130 - 149 decimal)
        +
        + See the definition of 'impression', 'sheet', and 'page'
        + in Section 2.
        + support bits starting:
        + { '00000000 00000000 00000000 00000000 20'H }
        +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

        pagesRequested(130),              Integer32 (-2..2147483647)
            INTEGER:  The number of logical pages requested by the job
            to be processed.

        pagesCompleted(131),              Integer32 (-2..2147483647)
            INTEGER:  The number of logical pages completed for this
            job so far.

            For implementations where multiple copies are produced by
            the interpreter with only a single pass over the data, the
            final value SHALL be equal to the value of the
            pagesRequested object.  For implementations where multiple
            copies are produced by the interpreter by processing the
            data for each copy, the final value SHALL be a multiple of
            the value of the pagesRequested object.

            NOTE - See the impressionsCompletedCurrentCopy and
            pagesCompletedCurrentCopy attributes for attributes that
            are reset on each document copy.

            NOTE - The pagesCompleted object can be used with the
            pagesRequested object to provide an indication of the
            relative progress of the job, provided that the
            multiplicative factor is taken into account for some
            implementations of multiple copies.
        pagesCompletedCurrentCopy(132),   Integer32 (-2..2147483647)
            INTEGER:  The number of logical pages completed for the
            current copy of the document so far.  This value SHALL be
            reset to 0 for each document in the job and for each
            document copy.

        +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
        + Sheet attributes (150 - 169 decimal)
        +
        + See the definition of 'impression', 'sheet', and 'page'
        + in Section 2.
        + support bits starting:
        + { '00000000 00000000 00000000 00000000 000002'H }
        +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

        sheetsRequested(150),             Integer32 (-2..2147483647)
            INTEGER:  The total number of medium sheets requested to be
            produced for this job.

            Unlike the jmJobKOctetsPerCopyRequested and
            jmJobImpressionsPerCopyRequested attributes, the
            sheetsRequested(150) attribute SHALL include the
            multiplicative factor contributed by the number of copies
            and so is the total number of sheets to be produced by the
            job, as opposed to the size of the document(s) submitted.

        sheetsCompleted(151),             Integer32 (-2..2147483647)
            INTEGER:  The total number of medium sheets that have
            completed marking and stacking for the entire job so far
            whether those sheets have been processed on one side or on
            both.

        sheetsCompletedCurrentCopy(152),  Integer32 (-2..2147483647)
            INTEGER:  The number of medium sheets that have completed
            marking and stacking for the current copy of a document in
            the job so far whether those sheets have been processed on
            one side or on both.

            The value of this attribute SHALL be 0 before the job
            starts processing and SHALL be reset to 1 after the first
            sheet of each document and document copy in the job is
            processed and stacked.
        +++++++++++++++++++++++++++++++++++++++++++++++++++++++++
        + Resources attributes (requested and consumed) (170 - 189)
        +
        + Pairs of these attributes can be used by monitoring
        + applications to show an indication of relative usage to
        + users, i.e., a 'thermometer'.
        + support bits starting:
        + { '00000000 00000000 00000000 00000000 00000000 0020'H }
        +++++++++++++++++++++++++++++++++++++++++++++++++++++++++

        mediumRequested(170),             JmMediumTypeTC
                                          AND/OR
                                          JmJobStringTC (SIZE(0..63))
            INTEGER:  MULTI-ROW:  The type
            AND/OR
            OCTETS:  MULTI-ROW:  the name of the medium that is
            required by the job.

            NOTE - The name (JmJobStringTC) values correspond to the
            name values of the prtInputMediaName object in the Printer
            MIB [print-mib] and the name, size, and input tray values
            of the IPP 'media' attribute [ipp-model].

        mediumConsumed(171),              Integer32 (-2..2147483647)
                                          AND
                                          JmJobStringTC (SIZE(0..63))
            INTEGER:  MULTI-ROW:  The number of sheets
            AND
            OCTETS:  MULTI-ROW:  the name of the medium that has been
            consumed so far whether those sheets have been processed on
            one side or on both.

            This attribute SHALL have both Integer32 and OCTET STRING
            (represented as  JmJobStringTC) values.

            NOTE - The name (JmJobStringTC) values correspond to the
            name values of the prtInputMediaName object in the Printer
            MIB [print-mib] and the name, size, and input tray values
            of the IPP 'media' attribute [ipp-model].

        colorantRequested(172),           Integer32 (-2..2147483647)
                                          AND/OR
                                          JmJobStringTC (SIZE(0..63))
            INTEGER:  MULTI-ROW:  The index (prtMarkerColorantIndex) in
            the Printer MIB[print-mib]
            AND/OR
            OCTETS:  MULTI-ROW:  the name of the colorant requested.

            NOTE - The name (JmJobStringTC) values correspond to the
            name values of the prtMarkerColorantValue object in the
            Printer MIB.  Examples are: red, blue.
        colorantConsumed(173),           Integer32 (-2..2147483647)
                                         AND/OR
                                         JmJobStringTC (SIZE(0..63))
            INTEGER:  MULTI-ROW:  The index (prtMarkerColorantIndex) in
            the Printer MIB[print-mib]
            AND/OR
            OCTETS:  MULTI-ROW:  the name of the colorant consumed.

            NOTE - The name (JmJobStringTC) values correspond to the
            name values of the prtMarkerColorantValue object in the
            Printer MIB.  Examples are: red, blue

        mediumTypeConsumed(174),         Integer32 (-2..2147483647)
                                         AND
                                         JmJobStringTC (SIZE(0..63))
            INTEGER:  MULTI-ROW:  The number of sheets of the indicated
            medium type that has been consumed so far whether those
            sheets have been processed on one side or on both
            AND
            OCTETS:  MULTI-ROW:  the name of that medium type.

            This attribute SHALL have both Integer32 and OCTET STRING
            (represented as JmJobStringTC) values.

            NOTE - The type name (JmJobStringTC) values correspond to
            the type name values of the prtInputMediaType object in the
            Printer MIB [print-mib].  Values are: 'stationery',
            'transparency', 'envelope', etc. These medium type names
            correspond to the enum values of JmMediumTypeTC used in the
            mediumRequested attribute.

        mediumSizeConsumed(175),         Integer32 (-2..2147483647)
                                         AND
                                         JmJobStringTC (SIZE(0..63))
            INTEGER:  MULTI-ROW:  The number of sheets of the indicated
            medium size that has been consumed so far whether those
            sheets have been processed on one side or on both
            AND
            OCTETS:  MULTI-ROW:  the name of that medium size.

            This attribute SHALL have both Integer32 and OCTET STRING
            (represented as JmJobStringTC) values.

            NOTE - The size name (JmJobStringTC) values correspond to
            the size name values in the Printer MIB [print-mib]
            Appendix B.  These size name values are also a subset of
            the keyword values defined by [ipp-model] for the 'media'
            Job Template attribute.  Values are:  'letter', 'a', 'iso-
            a4', 'jis-b4', etc.
        +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
        + Time attributes (set by server or device) (190 - 209 decimal)
        +
        + This section of attributes are ones that are set by the
        + server or device that accepts jobs.  Two forms of time are
        + provided.  Each form is represented in a separate attribute.
        + See section 3.1.2 and section 3.1.3 for the
        + conformance requirements for time attribute for agents and
        + monitoring applications, respectively.  The two forms are:
        +
        + 'DateAndTime' is an 8 or 11 octet binary encoded year,
        + month, day, hour, minute, second, deci-second with
        + optional offset from UTC.  See SNMPv2-TC [SMIv2-TC].
        +
        + NOTE: 'DateAndTime' is not printable characters; it is
        + binary.
        +
        + 'JmTimeStampTC' is the time of day measured in the number of
        + seconds since the system was booted.
        + support bits starting:
        + { '00000000 00000000 00000000 00000000 00000000 00000002'H }
        +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

        jobSubmissionToServerTime(190),   JmTimeStampTC
                                          AND/OR
                                          DateAndTime
            INTEGER:  Configuration 3 only:  The time
            AND/OR
            OCTETS:  the date and time that the job was submitted to
            the server (as distinguished from the device which uses
            jobSubmissionTime).

        jobSubmissionTime(191),           JmTimeStampTC
                                          AND/OR
                                          DateAndTime
            INTEGER:  Configurations 1, 2, and 3:  The time
            AND/OR
            OCTETS:  the date and time that the job was submitted to
            the server or device to which the agent is providing
            access.

        jobStartedBeingHeldTime(192),     JmTimeStampTC
                                          AND/OR
                                          DateAndTime
            INTEGER:  The time
            AND/OR
            OCTETS:  the date and time that the job last entered the
            pendingHeld state.  If the job has never entered the
            pendingHeld state, then the value SHALL be '0' or the
            attribute SHALL not be present in the table.
        jobStartedProcessingTime(193),    JmTimeStampTC
                                          AND/OR
                                          DateAndTime
            INTEGER:  The time
            AND/OR
            OCTETS:  the date and time that the job started processing.

        jobCompletionTime(194),           JmTimeStampTC
                                          AND/OR
                                          DateAndTime
            INTEGER:  The time
            AND/OR
            OCTETS:  the date and time that the job entered the
            completed, canceled, or aborted state.

        jobProcessingCPUTime(195)         Integer32 (-2..2147483647)
            UNITS     'seconds'
            INTEGER:  The amount of CPU time in seconds that the job
            has been in the processing state.  If the job enters the
            processingStopped state, that elapsed time SHALL not be
            included.  In other words, the jobProcessingCPUTime value
            SHOULD be relatively repeatable when the same job is
            processed again on the same device.
```

3.3.9 Job State Reason bit definitions

The JmJobStateReasons*N*TC (*N*=1..4) textual-conventions are used with the jmJobStateReasons1 object and jobStateReasonsN (*N*=2..4), respectively, to provide additional information regarding the current jmJobState object value. These values MAY be used with any job state or states for which the reason makes sense.

NOTE - While values cannot be added to the jmJobState object without impacting deployed clients that take actions upon receiving jmJobState values, it is the intent that additional JmJobStateReasons*N*TC enums can be defined and registered without impacting such deployed clients. In other words, the jmJobStateReasons1 object and jobStateReasons*N* attributes are intended to be extensible.

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

3.3.9.1 JmJobStateReasons1TC specification

The following standard values are defined (in hexadecimal) as *powers of two,* since multiple values MAY be used at the same time. For ease of understanding, the JmJobStateReasons1TC reasons are presented in the order in which the reasons are likely to occur (if implemented), starting with the 'jobIncoming' value and ending with the 'jobCompletedWithErrors' value.

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

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

        jobIncoming                     0x4
            The job has been accepted by the server or device, but the
            server or device is expecting (1) additional operations
            from the client to finish creating the job and/or (2) is
            accessing/accepting document data.

        submissionInterrupted           0x8
            The job was not completely submitted for some unforeseen
            reason, such as: (1) the server has crashed before the job
            was closed by the client, (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, (3) the client crashed or failed to close the job
            before the time-out period.

        jobOutgoing                     0x10
            Configuration 2 only:  The server is transmitting the job
            to the device.

        jobHoldSpecified                0x20
            The value of the job's jobHold(52) attribute is TRUE.  The
            job SHALL NOT be a candidate for processing until this
            reason is removed and there are no other reasons to hold
            the job.

        jobHoldUntilSpecified           0x40
            The value of the job's jobHoldUntil(53) attribute specifies
            a time period that is still in the future.  The job SHALL
            NOT be a candidate for processing until this reason is
            removed and there are no other reasons to hold the job.
```

jobProcessAfterSpecified 0x80

The value of the job's jobProcessAfterDateAndTime(51) attribute specifies a time that is still in the future. The job SHALL NOT be a candidate for processing until this reason is removed and there are no other reasons to hold the job.

resourcesAreNotReady 0x100

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. This condition MAY be detected when the job is accepted, or subsequently while the job is pending or processing, depending on implementation.

deviceStoppedPartly 0x200

One or more, but not all, of the devices to which the job is assigned are stopped. If all of the devices are stopped (or the only device is stopped), the deviceStopped reason SHALL be used.

deviceStopped 0x400

The device(s) to which the job is assigned is (are all) stopped.

jobInterpreting 0x800

The device to which the job is assigned is interpreting the document data.

jobPrinting 0x1000

The output device to which the job is assigned is marking media. This value is useful for servers and output devices which spend a great deal of time processing (1) when no marking is happening and then want to show that marking is now happening or (2) when the job is in the process of being canceled or aborted while the job remains in the processing state, but the marking has not yet stopped so that impression or sheet counts are still increasing for the job.

jobCanceledByUser 0x2000

The job was canceled by the owner of the job, i.e., by a user whose name is the same as the value of the job's jmJobOwner object, or by some other authorized end-user, such as a member of the job owner's security group.

jobCanceledByOperator 0x4000

The job was canceled by the operator, i.e., by a user who has been authenticated as having operator privileges (whether local or remote).

jobCanceledAtDevice 0x8000
The job was canceled by an unidentified local user, i.e., a user at a console at the device.

abortedBySystem 0x10000
The job (1) is in the process of being aborted, (2) has been aborted by the system and placed in the 'aborted' state, or (3) has been aborted by the system and placed in the 'pendingHeld' state, so that a user or operator can manually try the job again.

processingToStopPoint 0x20000
The requester has issued an operation to cancel or interrupt the job or the server/device has aborted the job, but the server/device is still performing some actions on the job until a specified stop point occurs or job termination/cleanup is completed.

This reason is recommended to be used in conjunction with the processing job state to indicate that the server/device is still performing some actions on the job while the job remains in the processing state. After all the job's resources consumed counters have stopped incrementing, the server/device moves the job from the processing state to the canceled or aborted job states.

serviceOffLine 0x40000
The service or document transform is off-line and accepting no jobs. All pending jobs are put into the pendingHeld state. This situation could be true if the service's or document transform's input is impaired or broken.

jobCompletedSuccessfully 0x80000
The job completed successfully.

jobCompletedWithWarnings 0x100000
The job completed with warnings.

jobCompletedWithErrors 0x200000
The job completed with errors (and possibly warnings too).

The following additional job state reasons have been added to represent job states that are in ISO DPA[iso-dpa] and other job submission protocols:

jobPaused	0x400000

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 agent SHALL remove the jobPaused values from the job’s jmJobStateReasons1 object and the job is eventually resumed at or near the point where the job was paused.

jobInterrupted	0x800000

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 device will automatically resume the interrupted job when the interrupting job completes.

jobRetained	0x1000000

The job is being retained by the server or device with all of the job’s document data (and submitted resources, such as fonts, logos, and forms, if any). Thus a client could issue an operation to the server or device to either (1) re-do the job (or a copy of the job) on the same server or device or (2) resubmit the job to another server or device. When a client could no longer re-do/resubmit the job, such as after the document data has been discarded, the agent SHALL remove the jobRetained value from the jmJobStateReasons1 object.

These bit definitions are the equivalent of a type 2 enum except that combinations of bits may be used together. See section 3.7.1.2. The remaining bits are reserved for future standardization and/or registration.

3.3.9.2 JmJobStateReasons2TC specification

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

```
        cascaded                            0x1
            An outbound gateway has transmitted all of the job's job
            and document attributes and data to another spooling
            system.

        deletedByAdministrator              0x2
            The administrator has deleted the job.

        discardTimeArrived                  0x4
            The job has been deleted due to the fact that the time
            specified by the job's job-discard-time attribute has
            arrived.

        postProcessingFailed                0x8
            The post-processing agent failed while trying to log
            accounting attributes for the job; therefore the job has
            been placed into the completed state with the jobRetained
            jmJobStateReasons1 object value for a system-defined period
            of time, so the administrator can examine it, resubmit it,
            etc.

        jobTransforming                     0x10
            The server/device is interpreting document data and
            producing another electronic representation.

        maxJobFaultCountExceeded            0x20
            The job has faulted several times and has exceeded the
            administratively defined fault count limit.

        devicesNeedAttentionTimeOut         0x40
            One or more document transforms that 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.

        needsKeyOperatorTimeOut             0x80
            One or more devices or document 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.
        jobStartWaitTimeOut                0x100
            The server/device has stopped the job 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
            pendingHeld state.

        jobEndWaitTimeOut                  0x200
            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 completed
            state.

        jobPasswordWaitTimeOut             0x400
            The server/device has stopped the job at the beginning of
            processing to await input of the job's password, but the
            password was not received within the site-settable time-out
            value.

        deviceTimedOut                     0x800
            A device that the job was using has not responded in a
            period specified by the device's site-settable attribute.

        connectingToDeviceTimeOut          0x1000
            The server is attempting to connect to 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.

        transferring                       0x2000
            The job is being transferred to a down stream server or
            downstream device.

        queuedInDevice                     0x4000
            The server/device has queued the job in a down stream
            server or downstream device.

        jobQueued                          0x8000
            The server/device has queued the document data.

        jobCleanup                         0x10000
            The server/device is performing cleanup activity as part of
            ending normal processing.
```

jobPasswordWait 0x20000

The server/device has selected the job to be next to process, but instead of assigning resources and starting the job processing, the server/device has transitioned the job to the pendingHeld state to await entry of a password (and dispatched another job, if there is one).

validating 0x40000

The server/device is validating the job *after* accepting the job.

queueHeld 0x80000

The operator has held the entire job set or queue.

jobProofWait 0x100000

The job has produced a single proof copy and is in the pendingHeld state waiting for the requester to issue an operation to release the job to print normally, obeying any job and document copy attributes that were originally submitted.

heldForDiagnostics 0x200000

The system is running intrusive diagnostics, so that all jobs are being held.

noSpaceOnServer 0x800000

There is no room on the server to store all of the job.

pinRequired 0x1000000

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.

exceededAccountLimit 0x2000000

The account for which this job is 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.

heldForRetry 0x4000000

The job encountered some errors that the server/device could not recover from with its normal retry procedures, but the error might not be encountered if the job is processed again in the future. Example cases are phone number busy or remote file system in-accessible. For such a situation, the server/device SHALL transition the job from the processing to the pendingHeld, rather than to the aborted state.

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

```
        canceledByShutdown                0x8000000
            The job was canceled because the server or device was
            shutdown before completing the job.

        deviceUnavailable                 0x10000000
            This job was aborted by the system because the device is
            currently unable to accept jobs.

        wrongDevice                       0x20000000
            This job was aborted by the system because the device is
            unable to handle this particular job; the spooler SHOULD
            try another device or the user should submit the job to
            another device.

        badJob                            0x40000000
            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.
```

These bit definitions are the equivalent of a type 2 enum except that combinations of them may be used together. See section 3.7.1.2.

3.3.9.3 JmJobStateReasons3TC specification

This textual-convention is used with the jobStateReasons3 attribute to provides additional information regarding the jmJobState object. The following standard values are defined (in hexadecimal) as *powers of two,* since multiple values may be used at the same time:

```
        jobInterruptedByDeviceFailure     0x1
            A device or the print system software that the job was
            using has failed while the job was processing.  The server
            or device is keeping the job in the pendingHeld state until
            an operator can determine what to do with the job.
```

These bit definitions are the equivalent of a type 2 enum except that combinations of them may be used together. See section 3.7.1.2. The remaining bits are reserved for future standardization and/or registration.

3.3.9.4 JmJobStateReasons4TC specification

This textual-convention is used with the jobStateReasons4 attribute to provides additional information regarding the jmJobState object. The following standard values are defined (in hexadecimal) as *powers of two,* since multiple values MAY be used at the same time.

None defined at this time.

These bit definitions are the equivalent of a type 2 enum except that combinations of them may be used together. See section 3.7.1.2. The remaining bits are reserved for future standardization and/or registration.

3.4 Monitoring Job Progress

There are a number of objects and attributes for monitoring the progress of a job. These objects and attributes count the number of K octets, impressions, sheets, and pages requested or completed. For impressions and sheets, "completed" means stacked, unless the implementation is unable to detect when each sheet is stacked, in which case stacked is approximated when processing of each sheet completes. There are objects and attributes for the overall job and for the current copy of the document currently being stacked. For the latter, the rate at which the various objects and attributes count depends on the sheet and document collation of the job.

Job Collation included sheet collation and document collation. Sheet collation is defined to be the ordering of sheets within a document copy. Document collation is defined to be ordering of document copies within a multi-document job. There are three types of job collation (see terminology definitions in Section 2):

1. uncollatedSheets(3) - No collation of the sheets within each document copy, i.e., each sheet of a document that is to produce multiple copies is replicated before the next sheet in the document is processed and stacked. If the device has an output bin collator, the uncollatedSheets(3) value may actually produce collated sheets as far as the user is concerned (in the output bins). However, when the job collation is the ’uncollatedSheets(3)’ value, job progress is indistinguishable to a monitoring application between a device that has an output bin collator and one that does not.

2. collatedDocuments(4) - Collation of the sheets within each document copy is performed within the printing device by making multiple passes over either the source or an intermediate representation of the document. In addition, when there are multiple documents per job, the i'th copy of each document is stacked before the j'th copy of each document, i.e., the documents are collated within each job copy. For example, if a job is submitted with documents, A and B, the job is made available to the end user as: A, B, A, B, ... The 'collatedDocuments(4)' value corresponds to the IPP [ipp-model] 'separate-documents-collated-copies' value of the "multiple-document-handling" attribute.

   If jobCopiesRequested or documentCopiesRequested = 1, then jobCollationType is defined as 4.

3. uncollatedDocuments(5) - Collation of the sheets within each document copy is performed within the printing device by making multiple passes over either the source or an intermediate representation of the document. In addition, when there are multiple documents per job, all copies of the first document in the job are stacked before the any copied of the next document in the job, i.e., the documents are uncollated within the job. For example, if a job is submitted with documents, A and B, the job is mad available to the end user as: A, A, ..., B, B, ... The 'uncollatedDocuments(5)' value corresponds to the IPP [ipp-model] 'separate-documents-uncollated-copies' value of the "multiple-document-handling" attribute.

Consider the following four variables that are used to monitor the progress of a job's impressions:

1. jmJobImpressionsCompleted - counts the total number of impressions stacked for the job

2. impressionsCompletedCurrentCopy - counts the number of impressions stacked for the current document copy

3. sheetCompletedCopyNumber - identifies the number of the copy for the current document being stacked where the first copy is 1.

4. sheetCompletedDocumentNumber - identifies the current document within the job that is being stacked where the first document in a job is 1. NOTE: this attribute SHOULD NOT be implemented for implementations that only support one document per job.

For each of the three types of job collation, a job with three copies of two documents (1, 2), where each document consists of 3 impressions, the four variables have the following values as each sheet is stacked for one-sided printing:

Job Collation Type = uncollatedSheets(3)

jmJobImpressions Completed	Impressions CompletedCurrent Copy	sheetCompleted CopyNumber	sheetCompleted DocumentNumber
0	0	0	0
1	1	1	1
2	1	2	1
3	1	3	1
4	2	1	1
5	2	2	1
6	2	3	1
7	3	1	1
8	3	2	1
9	3	3	1
10	1	1	2
11	1	2	2
12	1	3	2
13	2	1	2
14	2	2	2
15	2	3	2
16	3	1	2
17	3	2	2
18	3	3	2

Job Collation Type = collatedDocuments(4)

JmJobImpressions Completed	Impressions CompletedCurrent Copy	sheetCompleted CopyNumber	sheetCompleted DocumentNumber
0	0	0	0
1	1	1	1
2	2	1	1
3	3	1	1
4	1	1	2
5	2	1	2
6	3	1	2
7	1	2	1
8	2	2	1
9	3	2	1
10	1	2	2
11	2	2	2
12	3	2	2
13	1	3	1
14	2	3	1
15	3	3	1
16	1	3	2
17	2	3	2
18	3	3	2

Job Collation Type = uncollatedDocuments(5)

jmJobImpressions Completed	Impressions CompletedCurrent Copy	sheetCompleted CopyNumber	sheetCompleted DocumentNumber
0	0	0	0
1	1	1	1
2	2	1	1
3	3	1	1
4	1	2	1
5	2	2	1
6	3	2	1
7	1	3	1
8	2	3	1
9	3	3	1
10	1	1	2
11	2	1	2
12	3	1	2
13	1	2	2
14	2	2	2
15	3	2	2
16	1	3	2
17	2	3	2
18	3	3	2

3.5 Job Identification

There are a number of attributes that permit a user, operator or system administrator to identify jobs of interest, such as jobURI, jobName, jobOriginatingHost, etc. In addition, there is a jmJobSubmissionID object that is a text string table index. Being a table index allows a monitoring application to quickly locate and identify a particular job of interest that was submitted from a particular client by the user invoking the monitoring application without having to scan the entire job table. The Job Monitoring MIB needs to provide for identification of the job at both sides of the job submission process. The primary identification point is the client side. The jmJobSubmissionID allows the monitoring application to identify the job of interest from all the jobs currently "known" by the server or device. The value of jmJobSubmissionID can be assigned by either the client's local system or a downstream server or device. The point of assignment depends on the job submission protocol in use.

The server/device-side identifier, called the jmJobIndex object, SHALL be assigned by the SNMP Job Monitoring MIB agent when the server or device accepts the jobs from submitting clients. The jmJobIndex object allows the interested party to obtain all objects desired that relate to a particular job. See Section 3.2, entitled 'The Job Tables and the Oldest Active and Newest Active Indexes' for the specification of how the agent SHALL assign the jmJobIndex values.

The MIB provides a mapping table that maps each jmJobSubmissionID value to a corresponding jmJobIndex value generated by the agent, so that an application can determine the correct value for the jmJobIndex value for the job of interest in a single Get operation, given the Job Submission ID. See the jmJobIDGroup.

In some configurations there may be more than one application program that monitors the same job when the job passes from one network entity to another when it is submitted. See configuration 3. When there are multiple job submission IDs, each entity MAY supply an appropriate jmJobSubmissionID value. In this case there would be a separate entry in the jmJobSubmissionID table, one for each jmJobSubmissionID. All entries would map to the same jmJobIndex that contains the job data. When the job is deleted, it is up to the agent to remove all entries that point to the job from the jmJobSubmissionID table as well.

The jobName attribute provides a name that the user supplies as a job attribute with the job. The jobName attribute is not necessarily unique, even for one user, let alone across users.

3.5.1 The Job Submission ID specifications

This section specifies the formats for each of the registered Job Submission Ids. This format is used by the JmJobSubmissionIDTypeTC. Each job submission ID is a fixed-length, 48-octet printable US-ASCII [US-ASCII] coded character string containing no control characters, consisting of the following fields:

```
          octet  1:  The format letter identifying the format.  The US-
            ASCII characters '0-9', 'A-Z', and 'a-z' are assigned in
            order giving 62 possible formats.
          octets 2-40:  A 39-character, US-ASCII trailing SPACE filled
            field specified by the format letter, if the data is less
            than 39 ASCII characters.
          octets 41-48:  A sequential or random US-ASCII number to make
            the ID quasi-unique.
```

If the client does not supply a job submission ID in the job submission protocol, then the agent SHALL assign a job submission ID using any of the standard formats that are reserved for the agent. Clients SHALL not use formats that are reserved for agents and agents SHALL NOT use formats that are reserved for clients, in order to reduce conflicts in ID generation. See the description for which formats are reserved for clients or for agents.

Registration of additional formats may be done following the procedures described in Section 3.7.3.

The format values defined at the time of completion of this specification are:

```
Format
Letter  Description
------   ------------
'0' Job Owner generated by the server/device
octets 2-40:  The last 39 bytes of the jmJobOwner  object.
octets 41-48:  The US-ASCII 8-decimal-digit sequential number
    assigned by the agent.
This format is reserved for agents.

NOTE - Clients wishing to use a job submission ID that
    incorporates the job owner, SHALL use format '8', not
    format '0'.

'1' Job Name
octets 2-40:  The last 39 bytes of the jobName attribute.
octets 41-48:  The US-ASCII 8-decimal-digit random number
    assigned by the client.
This format is reserved for clients.

'2' Client MAC address
octets 2-40:  The client MAC address: in hexadecimal with each
    nibble of the 6 octet address being '0'-'9' or 'A' - 'F'
    (uppercase only). Most significant octet first.
octets 41-48:  The US-ASCII 8-decimal-digit sequential number
    assigned by the client.
This format is reserved for clients.

'3' Client URL
octets 2-40:  The last 39 bytes of the client URL [URI-spec].
octets 41-48:  The US-ASCII 8-decimal-digit sequential number
    assigned by the client.
This format is reserved for clients.

'4' Job URI
octets 2-40:  The last 39 bytes of the URI [URI-spec] assigned
    by the server or device to the job when the job was
    submitted for processing.
octets 41-48:  The US-ASCII 8-decimal-digit sequential number
    assigned by the agent.
This format is reserved for agents.

'5' POSIX User Number
octets 2-40:  The last 39 bytes of a user number, such as POSIX
    user number.
octets 41-48:  The US-ASCII 8-decimal-digit sequential number
    assigned by the client.
```

```
            This format is reserved for clients.

            ’6’ User Account Number
            octets 2-40:  The last 39 bytes of the user account number.
            octets 41-48:  The US-ASCII 8-decimal-digit sequential number
                assigned by the client.
            This format is reserved for clients.

            ’7’ DTMF Incoming FAX routing number
            octets 2-40:  The last 39 bytes of the DTMF incoming FAX
                routing number.
            octets 41-48:  The US-ASCII 8-decimal-digit sequential number
                assigned by the client.
            This format is reserved for clients.

            ’8’ Job Owner supplied by the client
            octets 2-40:  The last 39 bytes of the job owner name (that the
                agent returns in the jmJobOwner object).
            octets 41-48:  The US-ASCII 8-decimal-digit sequential number
                assigned by the client.
            This format is reserved for clients.  See format ’0’ which is
                reserved for agents.

            ’9’ Host Name
            octets 2-40:  The last 39 bytes of the host name with trailing
                SPACES that submitted the job to this server/device using a
                protocol, such as LPD [RFC1179] which includes the host
                name in the job submission protocol.
            octets 41-48:  The US-ASCII 8-decimal-digit leading zero
                representation of the job id generated by the submitting
                server (configuration 3) or the client (configuration 1 and
                2), such as in the LPD protocol.
            This format is reserved for clients.

            ’A’ AppleTalk Protocol
            octets 2-40:  Contains the AppleTalk printer name, with the
                first character of the name in octet 2.  AppleTalk printer
                names are a maximum of 31 characters.  Any unused portion
                of this field shall be filled with spaces.
            octets 41-48:  ’00000XXX’, where ’XXX’ is the 3-digit US-ASCII
                decimal representation of the Connection Id.
            This format is reserved for agents.
```

```
        'B' NetWare PServer
        octets 2-40:  Contains the Directory Path Name as recorded by
            the Novell File Server in the queue directory.  If the
            string is less than 40 octets, the left-most character in
            the string shall appear in octet position 2.  Otherwise,
            only the last 39 bytes shall be included.  Any unused
            portion of this field shall be filled with spaces.
        octets 41-48:  '000XXXXX'  The US-ASCII representation of the
            Job Number as per the NetWare File Server Queue Management
            Services.
        This format is reserved for agents.

        'C' Server Message Block protocol (SMB)
        octets 2-40:  Contains a decimal (US-ASCII coded)
            representation of the 16 bit SMB Tree Id field, which
            uniquely identifies the connection that submitted the job
            to the printer.  The most significant digit of the numeric
            string shall be placed in octet position 2.  All unused
            portions of this field shall be filled with spaces.  The
            SMB Tree Id has a maximum value of 65,535.
        octets 41-48:  The US-ASCII 8-decimal-digit leading zero
            representation of the File Handle returned from the device
            to the client in response to a Create Print File command.
        This format is reserved for agents.

        'D' Transport Independent Printer/System Interface (TIP/SI)
        octets 2-40:  Contains the Job Name from the Job Control-Start
            Job (JC-SJ) command.  If the Job Name portion is less than
            40 octets, the left-most character in the string shall
            appear in octet position 2.  Any unused portion of this
            field shall be filled with spaces.  Otherwise, only the
            last 39 bytes shall be included.
        octets 41-48:  The US-ASCII 8-decimal-digit leading zero
            representation of the jmJobIndex assigned by the agent.
        This format is reserved for agents, since the agent supplies
            octets 41-48, though the client supplies the job name.  See
            format '1' reserved to clients to submit job name ids in
            which they supply octets 41-48.

        'E' IPDS on the MVS or VSE platform

        octets 2-40:  Contains bytes 2-27 of the XOH Define Group
            Boundary Group ID triplet. Octet position 2 MUST carry the
            value x'01'.  Bytes 28-40 MUST be filled with spaces.
        octets 41-48: The US-ASCII 8-decimal-digit leading zero
            representation of the jmJobIndex assigned by the agent.
        This format is reserved for agents, since the agent supplies
            octets 41-48, though the client supplies the job name.
        'F' IPDS on the VM platform
        octets 2-40:  Contains bytes 2-31 of the XOH Define Group
            Boundary Group ID triplet. Octet position 2 MUST carry the
            value x'02'.  Bytes 32-40 MUST be filled with spaces.
        octets 41-48: The US-ASCII 8-decimal-digit leading zero
            representation of the jmJobIndex assigned by the agent.
        This format is reserved for agents, since the agent supplies
            octets 41-48, though the client supplies the file name.

        'G' IPDS on the OS/400 platform
        octets 2-40:  Contains bytes 2-36 of the XOH Define Group
            Boundary Group ID triplet.  Octet position 2 MUST carry the
            value x'03'.  Bytes 37-40 MUST be filled with spaces.
        octets 41-48: The US-ASCII 8-decimal-digit leading zero
            representation of the jmJobIndex assigned by the agent.
        This format is reserved for agents, since the agent supplies
            octets 41-48, though the client supplies the job name.
```

NOTE - the job submission id is only intended to be unique between a limited set of clients for a limited duration of time, namely, for the life time of the job in the context of the server or device that is processing the job. Some of the formats include something that is unique per client and a random number so that the same job submitted by the same client will have a different job submission id. For other formats, where part of the id is guaranteed to be unique for each client, such as the MAC address or URL, a sequential number SHOULD suffice for each client (and may be easier for each client to manage). Therefore, the length of the job submission id has been selected to reduce the probability of collision to an extremely low number, but is not intended to be an absolute guarantee of uniqueness. None-the-less, collisions are remotely possible, but without bad consequences, since this MIB is intended to be used only for monitoring jobs, not for controlling and managing them.

3.6 Internationalization Considerations

This section describes the internationalization considerations included in this MIB.

3.6.1 Text generated by the server or device

There are a few objects and attributes generated by the server or device that SHALL be represented using the Universal Multiple-Octet Coded Character Set (UCS) [ISO-10646]. These objects and attributes are always supplied (if implemented) by the agent, not by the job submitting client:

1. jmGeneralJobSetName object
2. processingMessage(6) attribute
3. physicalDevice(32) (name value) attribute

The character encoding scheme for representing these objects and attributes SHALL be UTF-8 as REQUIRED by RFC 2277 [RFC2277]. The 'JmUTF8StringTC' textual convention is used to indicate UTF-8 text strings.

NOTE - For strings in 7-bit US-ASCII, there is no impact since the UTF-8 representation of 7-bit ASCII is identical to the US-ASCII [US-ASCII] encoding.

The text contained in the processingMessage(6) attribute is generated by the server/device. The natural language for the processingMessage(6) attribute is identified by the processingMessageNaturalLangTag(7) attribute. The processingMessageNaturalLangTag(7) attribute uses the JmNaturalLanguageTagTC textual convention which SHALL conform to the language tag mechanism specified in RFC 1766 [RFC-1766]. The JmNaturalLanguageTagTC value is the same as the IPP [IPP-model] 'naturalLanguage' attribute syntax. RFC 1766 specifies that a US-ASCII string consisting of the natural language followed by an optional country field. Both fields use the same two-character codes from ISO 639 [ISO-639] and ISO 3166 [ISO-3166], respectively, that are used in the Printer MIB for identifying language and country.

Examples of the values of the processingMessageNaturalLangTag(7) attribute include:

1. 'en' for English
2. 'en-us' for US English
3. 'fr' for French
4. 'de' for German

3.6.2 Text supplied by the job submitter

All of the objects and attributes represented by the 'JmJobStringTC' textual-convention are either (1) supplied in the job submission protocol by the client that submits the job to the server or device or (2) are defaulted by the server or device if the job submitting client does not supply values. The agent SHALL represent these objects and attributes in the MIB either (1) in the coded character set as they were submitted or (2) MAY convert the coded character set to another coded character set or encoding scheme. In any case, the resulting coded character set representation SHOULD be UTF-8 [UTF-8], but SHALL be one in which the code positions from 0 to 31 is not used, 32 to 127 is US-ASCII [US-ASCII], 127 is not unused, and the remaining code positions 128 to 255 represent single-byte or multi-byte graphic characters structured according to ISO 2022 [ISO-2022] or are unused.

The coded character set SHALL be one of the ones registered with IANA [IANA] and SHALL be identified by the jobCodedCharSet attribute in the jmJobAttributeTable for the job. If the agent does not know what coded character set was used by the job submitting client, the agent SHALL either (1) return the 'unknown(2)' value for the jobCodedCharSet attribute or (2) not return the jobCodedCharSet attribute for the job.

Examples of coded character sets which meet this criteria for use as the value of the jobCodedCharSet job attribute are: US-ASCII [US-ASCII], ISO 8859-1 (Latin-1) [ISO-8859-1], any ISO 8859-n, HP Roman8, IBM Code Page 850, Windows Default 8-bit set, UTF-8 [UTF-8], US-ASCII plus JIS X0208-1990 Japanese [JIS X0208], US-ASCII plus GB2312-1980 PRC Chinese [GB2312]. See the IANA registry of coded character sets [IANA charsets].

Examples of coded character sets which do not meet this criteria are: national 7-bit sets conforming to ISO 646 (except US-ASCII), EBCDIC, and ISO 10646 (Unicode) [ISO-10646]. In order to represent Unicode characters, the UTF-8 [UTF-8] encoding scheme SHALL be used which has been assigned the MIBenum value of '106' by IANA.

The jobCodedCharSet attribute uses the imported 'CodedCharSet' textual-convention from the Printer MIB [printmib].

The natural language for attributes represented by the textual-convention JmJobStringTC is identified either (1) by the jobNaturalLanguageTag(9) attribute or is keywords in US-English (as in IPP). A monitoring application SHOULD attempt to localize keywords into the language of the user by means of some lookup mechanism. If the keyword value is not known to the monitoring application, the monitoring application SHOULD assume that the value is in the natural language specified by the job's jobNaturalLanguageTag(9) attribute and SHOULD present the value to its user as is. The jobNaturalLanguageTag(9) attribute value SHALL have the same syntax and semantics as the processingMessageNaturalLangTag(7) attribute, except that the jobNaturalLanguageTag(9) attribute identifies the natural language of attributes supplied by the job submitter instead of the natural language of the processingMessage(6) attribute. See Section 3.6.1.

3.6.3 'DateAndTime' for representing the date and time

This MIB also contains objects that are represented using the DateAndTime textual convention from SMIv2 [SMIv2-TC]. The job management application SHALL display such objects in the locale of the user running the monitoring application.

3.7 IANA and PWG Registration Considerations

This MIB does not require any additional registration schemes for IANA, but does depend on registration schemes that other Internet standards track specifications have set up. The names of these IANA registration assignments under the /in-notes/iana/assignments/ path:

1. printer-language-numbers - used as enums in the documentFormat(38) attribute

2. media-types - uses as keywords in the documentFormat(38) attribute

3. character-sets - used as enums in the jobCodedCharSet(8) attribute

The Printer Working Group (PWG) will handle registration of additional enums after approving this standard, according to the procedures described in this section:

3.7.1 PWG Registration of enums

This specification uses textual conventions to define enumerated values (enums) and bit values. Enumerations (enums) and bit values are sets of symbolic values defined for use with one or more objects or attributes. All enumeration sets and bit value 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 defined at the beginning of the MIB module specification.

The PWG has defined several type of enumerations for use in the Job Monitoring MIB and the Printer MIB[print-mib]. These types 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:

3.7.1.1 Type 1 enumerations

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.

There are no type 1 enums in the current draft.

3.7.1.2 Type 2 enumerations

Type 2 enumeration: An initial set of values are defined in the Job Monitoring MIB specification. Additional enumerated values are registered with the PWG.

The following type 2 enums are contained in the current draft :

1. JmUTF8StringTC
2. JmJobStringTC
3. JmNaturalLanguageTagTC
4. JmTimeStampTC
5. JmFinishingTC [same enum values as IPP "finishing" attribute]
6. JmPrintQualityTC [same enum values as IPP "print-quality" attribute]
7. JmTonerEconomyTC
8. JmMediumTypeTC
9. JmJobSubmissionIDTypeTC
10. JmJobCollationTypeTC
11. JmJobStateTC [same enum values as IPP "job-state" attribute]
12. JmAttributeTypeTC

For those textual conventions that have the same enum values as the indicated IPP Job attribute are simultaneously registered by the PWG for use with IPP [ipp-model] and the Job Monitoring MIB.

3.7.1.3 Type 3 enumeration

Type 3 enumeration: An initial set of values are defined in the Job Monitoring MIB specification. Additional enumerated values are registered through the PWG without PWG review.

There are no type 3 enums in the current draft.

3.7.2 PWG Registration of type 2 bit values

This draft contains the following type 2 bit value textual-conventions:

1. JmJobServiceTypesTC
2. JmJobStateReasons1TC
3. JmJobStateReasons2TC
4. JmJobStateReasons3TC
5. JmJobStateReasons4TC

These textual-conventions are defined as bits in an Integer so that they can be used with SNMPv1 SMI. The jobStateReasons*N* (*N*=1..4) attributes are defined as bit values using the corresponding JmJobStateReasons*N*TC textual-conventions.

The registration of JmJobServiceTypesTC and JmJobStateReasons*N*TC bit values follow the procedures for a type 2 enum as specified in Section 3.7.1.2.

3.7.3 PWG Registration of Job Submission Id Formats

In addition to enums and bit values, this specification assigns a single ASCII digit or letter to various job submission ID formats. See the JmJobSubmissionIDTypeTC textual-convention and the object. The registration of JobSubmissionID format numbers follows the procedures for a type 2 enum as specified in Section 3.7.1.2.

3.7.4 PWG Registration of MIME types/sub-types for document-formats

The documentFormat(38) attribute has MIME type/sub-type values for indicating document formats which IANA registers as "media type" names. The values of the documentFormat(38) attribute are the same as the corresponding Internet Printing Protocol (IPP) "document-format" Job attribute values [ipp-model].

3.8 Security Considerations

3.8.1 Read-Write objects

All objects are read-only, greatly simplifying the security considerations. If another MIB augments this MIB, that MIB might accept SNMP Write operations to objects in that MIB whose effect is to modify the values of read-only objects in this MIB. However, that MIB SHALL have to support the required access control in order to achieve security, not this MIB.

3.8.2 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 jmJobKOctetsPerCopyRequested and jmJobKOctetsProcessed objects, so that a user can tell how busy a printer is. Other sites MAY allow all unprivileged users to see all objects of all jobs. This MIB does not require, nor does it specify how, such restrictions would be implemented. A monitoring application SHOULD enforce the site security policy with respect to returning information to an unprivileged end user that is using the monitoring application to monitor jobs that do not belong to that user, i.e., the jmJobOwner object in the jmJobTable does not match the user’s user name.

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

3.9 Notifications

This MIB does not specify any notifications. For simplicity, management applications are expected to poll for status. The jmGeneralJobPersistence and jmGeneralAttributePersistence objects assist an application to determine the polling rate. The resulting network traffic is not expected to be significant.

4 MIB specification

The following pages constitute the actual Job Monitoring MIB.

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

IMPORTS
     MODULE-IDENTITY, OBJECT-TYPE, enterprises,
     Integer32                                        FROM SNMPv2-SMI
     TEXTUAL-CONVENTION                               FROM SNMPv2-TC
     MODULE-COMPLIANCE, OBJECT-GROUP                  FROM SNMPv2-CONF;
     -- The following textual-conventions are needed to implement
     -- certain attributes, but are not needed to compile this MIB.
     -- They are provided here for convenience:
     -- hrDeviceIndex                         FROM HOST-RESOURCES-MIB
     -- DateAndTime                           FROM SNMPv2-TC
     -- PrtInterpreterLangFamilyTC,
     -- CodedCharSet                          FROM Printer-MIB

-- Use the enterprises arc assigned to the PWG which is pwg(2699).
-- Group all PWG mibs under mibs(1).

jobmonMIB MODULE-IDENTITY
    LAST-UPDATED "9902200000Z"
    ORGANIZATION "Printer Working Group (PWG)"
    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

        Send questions and comments to the Printer Working Group (PWG)
        using the Job Monitoring Project (JMP) Mailing List:
        jmp@pwg.org

        For further information, including how to subscribe to the
        jmp mailing list, access the PWG web page under 'JMP':

            http://www.pwg.org/

        Implementers of this specification are encouraged to join the
        jmp mailing list in order to participate in discussions on any
        clarifications needed and registration proposals being reviewed
        in order to achieve consensus."
    DESCRIPTION
        "The MIB module for monitoring job in servers, printers, and
        other devices.

        Version: 2.0"
    ::= { enterprises pwg(2699)  mibs(1)  jobmonMIB(1) }
```

```
-- Textual conventions for this MIB module

JmUTF8StringTC ::= TEXTUAL-CONVENTION
    DISPLAY-HINT "255a"
    STATUS      current
    DESCRIPTION
        "To facilitate internationalization, this TC represents
        information taken from the ISO/IEC IS 10646-1 character set,
        encoded as an octet string using the UTF-8 character encoding
        scheme.

        See section 3.6.1, entitled: 'Text generated by the server or
        device'."
    SYNTAX      OCTET STRING (SIZE (0..63))




JmJobStringTC ::= TEXTUAL-CONVENTION
    STATUS      current
    DESCRIPTION
        "To facilitate internationalization, this TC represents
        information using any coded character set registered by IANA as
        specified in section 3.7.  While it is recommended that the
        coded character set be UTF-8 [UTF-8], the actual coded
        character set SHALL be indicated by the value of the
        jobCodedCharSet(8) attribute for the job.

        See section 3.6.2, entitled: 'Text supplied by the job
        submitter'."
    SYNTAX      OCTET STRING (SIZE (0..63))




JmNaturalLanguageTagTC  ::= TEXTUAL-CONVENTION
    STATUS      current
    DESCRIPTION
        "An IETF RFC 1766-compliant 'language tag', with zero or more
        sub-tags that identify a natural language.  While RFC 1766
        specifies that the US-ASCII values are case-insensitive, this
        MIB specification requires that all characters SHALL be lower
        case in order to simplify comparing by management applications.

        See section 3.6.1, entitled: 'Text generated by the server or
        device' and section 3.6.2, entitled: 'Text supplied by the job
        submitter'."
    SYNTAX      OCTET STRING (SIZE (0..63))
```

```
JmTimeStampTC ::= TEXTUAL-CONVENTION
    STATUS      current
    DESCRIPTION
        "The simple time at which an event took place.  The units are
        in seconds since the system was booted.

        NOTE - JmTimeStampTC is defined in units of seconds, rather
        than 100ths of seconds, so as to be simpler for agents to
        implement (even if they have to implement the 100ths of a
        second to comply with implementing sysUpTime in MIB-II[mib-
        II].)

        NOTE - JmTimeStampTC is defined as an Integer32 so that it can
        be used as a value of an attribute, i.e., as a value of the
        jmAttributeValueAsInteger object.  The TimeStamp textual-
        convention defined in SNMPv2-TC [SMIv2-TC] is defined as an
        APPLICATION 3 IMPLICIT INTEGER tag, not an Integer32 which is
        defined in SNMPv2-SMI [SMIv2-TC] as UNIVERSAL 2 IMPLICIT
        INTEGER, so cannot be used in this MIB as one of the values of
        jmAttributeValueAsInteger."
    SYNTAX      INTEGER (0..2147483647)




JmJobSourcePlatformTypeTC ::= TEXTUAL-CONVENTION
    STATUS      current
    DESCRIPTION
        "The source platform type that can submit jobs to servers or
        devices in any of the 3 configurations.

        This is a type 2 enumeration.  See Section 3.7.1.2.  See also
        IANA operating-system-names registry."
    SYNTAX      INTEGER {
         other(1),
         unknown(2),
         sptUNIX(3),           -- UNIX
         sptOS2(4),            -- OS/2
         sptPCDOS(5),          -- DOS
         sptNT(6),             -- NT
         sptMVS(7),            -- MVS
         sptVM(8),             -- VM
         sptOS400(9),          -- OS/400
         sptVMS(10),           -- VMS
         sptWindows(11),       -- Windows
         sptNetWare(12)        -- NetWare
    }
```

```
JmFinishingTC ::= TEXTUAL-CONVENTION
    STATUS      current
    DESCRIPTION
        "The type of finishing operation.

        These values are the same as the enum values of the IPP
        'finishings' attribute.  See Section 3.7.1.2.

        other(1),
            Some other finishing operation besides one of the specified
            or registered values.

        unknown(2),
            The finishing is unknown.

        none(3),
            Perform no finishing.

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

        punch(5),
            Holes are required in the finished document. The exact
            number and placement of the holes is site-defined.  The
            punch specification MAY be satisfied (in a site- and
            implementation-specific manner) either by
            drilling/punching, or by substituting pre-drilled media.

        cover(6),
            Select a non-printed (or pre-printed) cover for the
            document. This does not supplant the specification of a
            printed cover (on cover stock medium) by the document
            itself.

        bind(7)
            Binding is to be applied to the document; the type and
            placement of the binding is product-specific.

        This is a type 2 enumeration.  See Section 3.7.1.2."
    SYNTAX      INTEGER {
        other(1),
        unknown(2),
        none(3),
        staple(4),
        punch(5),
        cover(6),
        bind(7)
    }
```

```
JmPrintQualityTC ::= TEXTUAL-CONVENTION
    STATUS      current
    DESCRIPTION
        "Print quality settings.

        These values are the same as the enum values of the IPP 'print-
        quality' attribute.  See Section 3.7.1.2.

        This is a type 2 enumeration.  See Section 3.7.1.2."
    SYNTAX      INTEGER {
         other(1),    -- Not one of the specified or registered
                      -- values.
         unknown(2),  -- The actual value is unknown.
         draft(3),    -- Lowest quality available on the printer.
         normal(4),   -- Normal or intermediate quality on the
                      -- printer.
         high(5)      -- Highest quality available on the printer.
    }



JmPrinterResolutionTC ::= TEXTUAL-CONVENTION
    STATUS      current
    DESCRIPTION
        "Printer resolutions.

        Nine octets consisting of two 4-octet SIGNED-INTEGERs followed
        by a SIGNED-BYTE.  The values are the same as those specified
        in the Printer MIB [printmib]. The first SIGNED-INTEGER
        contains the value of prtMarkerAddressabilityXFeedDir.  The
        second SIGNED-INTEGER contains the value of
        prtMarkerAddressabilityFeedDir.  The SIGNED-BYTE contains the
        value of prtMarkerAddressabilityUnit.

        Note: the latter value is either 3 (tenThousandsOfInches) or 4
        (micrometers) and the addressability is in 10,000 units of
        measure. Thus the SIGNED-INTEGERs represent integral values in
        either dots-per-inch or dots-per-centimeter.

        The syntax is the same as the IPP 'printer-resolution'
        attribute.  See Section 3.7.1.2."
    SYNTAX      OCTET STRING (SIZE(9))
```

```
JmTonerEconomyTC ::= TEXTUAL-CONVENTION
    STATUS      current
    DESCRIPTION
        "Toner economy settings.

        This is a type 2 enumeration.  See Section 3.7.1.2."
    SYNTAX      INTEGER {
         unknown(2),    --  unknown.
         off(3),        --  Off.  Normal.  Use full toner.
         on(4)          --  On.  Use less toner than normal.
    }



JmBooleanTC ::= TEXTUAL-CONVENTION
    STATUS      current
    DESCRIPTION
        "Boolean true or false value.

        This is a type 2 enumeration.  See Section 3.7.1.2."
    SYNTAX      INTEGER {
         unknown(2),    --  unknown.
         false(3),      --  FALSE.
         true(4)        --  TRUE.
    }



JmMediumTypeTC ::= TEXTUAL-CONVENTION
    STATUS      current
    DESCRIPTION
        "Identifies the type of medium.

        other(1),
            The type is neither one of the values listed in this
            specification nor a registered value.

        unknown(2),
            The type is not known.

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

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

        envelope(5),
            Envelopes that can be used for conventional mailing
            purposes.
        envelopePlain(6),
            Envelopes that are not preprinted and have no windows.

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

        continuousLong(8),
            Continuously connected sheets of an opaque material
            connected along the long edge.

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

        tabStock(10),
            Media with tabs.

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

        labels(12),
            Label-stock.

        multiLayer(13)
            Form medium composed of multiple layers which are pre-
            attached to one another, e.g. for use with impact printers.

        This is a type 2 enumeration.  See Section 3.7.1.2.  These enum
        values correspond to the keyword name strings of the
        prtInputMediaType object in the Printer MIB [print-mib].  There
        is no printer description attribute in IPP/1.0 that represents
        these values."
    SYNTAX      INTEGER {
        other(1),
        unknown(2),
        stationery(3),
        transparency(4),
        envelope(5),
        envelopePlain(6),
        envelopeWindow(7),
        continuousLong(8),
        continuousShort(9),
        tabStock(10),
        multiPartForm(11),
        labels(12),
        multiLayer(13)
    }
```

```
JmJobCollationTypeTC ::= TEXTUAL-CONVENTION
    STATUS      current
    DESCRIPTION
        "This value is the type of job collation.  Implementations that
        don’t support multiple documents or don’t support multiple
        copies SHALL NOT support the uncollatedDocuments(5) value.

        This is a type 2 enumeration.  See Section 3.7.1.2. See also
        Section 3.4, entitled ’Monitoring Job Progress’."
    SYNTAX      INTEGER {
        other(1),
        unknown(2),
        uncollatedSheets(3),    -- sheets within each document copy
                                -- are not collated: 1 1 ..., 2 2 ...,
                                -- No corresponding value of IPP
                                -- "multiple-document-handling"
        collatedDocuments(4),   -- internal collated sheets,
                                -- documents: A, B, A, B, ...
                                -- Corresponds to IPP "multiple-
                                -- document-handling"=’separate-
                                -- documents-collated-copies’
        uncollatedDocuments(5)  -- internal collated sheets,
                                -- documents: A, A, ..., B, B, ...
                                -- Corresponds to IPP "multiple-
                                -- document-handling"=’separate-
                                -- documents-uncollated-copies’
    }


JmJobSubmissionIDTypeTC ::= TEXTUAL-CONVENTION
    STATUS      current
    DESCRIPTION
        "Identifies the format type of a job submission ID.

        Each job submission ID is a fixed-length, 48-octet printable
        US-ASCII [US-ASCII] coded character string containing no
        control characters, consisting of the fields defined in section
        3.5.1.

        This is like a type 2 enumeration.  See section 3.7.3."
    SYNTAX    OCTET STRING(SIZE(1)) -- ASCII ’0’-’9’, ’A’-’Z’, ’a’-’z’
```

```
JmJobStateTC ::= TEXTUAL-CONVENTION
    STATUS      current
    DESCRIPTION
        "The current state of the job (pending, processing, completed,
        etc.).  The following figure shows the normal job state
        transitions:

                                                    +----> canceled(7)
                                                   /
    +---> pending(3) -------> processing(5) ------+------> completed(9)
    |         ^                      ^             \
--->+         |                      |              +----> aborted(8)
    |         v                      v             /
    +---> pendingHeld(4)  processingStopped(6) ---+

                Figure 4 - Normal Job State Transitions

        Normally a job progresses from left to right.  Other state
        transitions are unlikely, but are not forbidden.  Not shown are
        the transitions to the canceled state from the pending,
        pendingHeld, and processingStopped states.

        Jobs in the pending, processing, and processingStopped states
        are called 'active', while jobs in the pendingHeld, canceled,
        aborted, and completed states are called 'inactive'.  Jobs
        reach one of the three terminal states: completed, canceled, or
        aborted, after the jobs have completed all activity, and all
        MIB objects and attributes have reached their final values for
        the job.

        These values are the same as the enum values of the IPP 'job-
        state' job attribute.  See Section 3.7.1.2.

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

        pending(3),
            The job is a candidate to start processing, but is not yet
            processing.

        pendingHeld(4),
            The job is not a candidate for processing for any number of
            reasons but will return to the pending state as soon as the
            reasons are no longer present.  The job's
            jmJobStateReasons1 object and/or jobStateReasonsN (N=2..4)
            attributes SHALL indicate why the job is no longer a
            candidate for processing.  The reasons are represented as
            bits in the jmJobStateReasons1 object and/or
            jobStateReasonsN (N=2..4) attributes.  See the
```

JmJobStateReasons*N*TC (*N*=1..4) textual convention for the specification of each reason.

processing(5),

One or more of:

1. the job is using, or is attempting to use, one or more purely software processes that are analyzing, creating, or interpreting a PDL, etc.,

2. the job is using, or is attempting to use, one or more hardware devices that are interpreting a PDL, making marks on a medium, and/or performing finishing, such as stapling, etc., OR

3. (configuration 2) the server has made the job ready for printing, but the output device is not yet printing it, either because the job hasn't reached the output device or because the job is queued in the output device or some other spooler, awaiting the output device to print it.

When the job is in the processing state, the entire job state includes the detailed status represented in the device MIB indicated by the hrDeviceIndex value of the job's physicalDevice attribute, if the agent implements such a device MIB.

Implementations MAY, though they NEED NOT, include additional values in the job's jmJobStateReasons1 object to indicate the progress of the job, such as adding the jobPrinting value to indicate when the device is actually making marks on a medium and/or the processingToStopPoint value to indicate that the server or device is in the process of canceling or aborting the job.

processingStopped(6),

The job has stopped while processing for any number of reasons and will return to the processing state as soon as the reasons are no longer present.

The job's jmJobStateReasons1 object and/or the job's jobStateReasons*N* (*N*=2..4) attributes MAY indicate why the job has stopped processing. For example, if the output device is stopped, the deviceStopped value MAY be included in the job's jmJobStateReasons1 object.

NOTE - When an output device is stopped, the device usually indicates its condition in human readable form at the device. The management application can obtain more complete device status remotely by querying the appropriate device MIB using the job's deviceIndex attribute(s), if the agent implements such a device MIB

```
        canceled(7),
            A client has canceled the job and the server or device has
            completed canceling the job AND all MIB objects and
            attributes have reached their final values for the job.
            While the server or device is canceling the job, the job's
            jmJobStateReasons1 object SHOULD contain the
            processingToStopPoint value and one of the canceledByUser,
            canceledByOperator, or canceledAtDevice values.  The
            canceledByUser, canceledByOperator, or canceledAtDevice
            values remain while the job is in the canceled state.

        aborted(8),
            The job has been aborted by the system, usually while the
            job was in the processing or processingStopped state and
            the server or device has completed aborting the job AND all
            MIB objects and attributes have reached their final values
            for the job.  While the server or device is aborting the
            job, the job's jmJobStateReasons1 object MAY contain the
            processingToStopPoint and abortedBySystem values.  If
            implemented, the abortedBySystem value SHALL remain while
            the job is in the aborted state.

        completed(9)
            The job has completed successfully or with warnings or
            errors after processing and all of the media have been
            successfully stacked in the appropriate output bin(s) AND
            all MIB objects and attributes have reached their final
            values for the job.  The job's jmJobStateReasons1 object
            SHOULD contain one of: completedSuccessfully,
            completedWithWarnings, or completedWithErrors values.

        This is a type 2 enumeration.  See Section 3.7.1.2."
    SYNTAX      INTEGER {
        unknown(2),
        pending(3),
        pendingHeld(4),
        processing(5),
        processingStopped(6),
        canceled(7),
        aborted(8),
        completed(9)
    }
```

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

        NOTE - The enum assignments are grouped logically with values
        assigned in groups of 20, so that additional values may be
        registered in the future and assigned a value that is part of
        their logical grouping.

        Values in the range 2**30 to 2**31-1 are reserved for private
        or experimental usage.  This range corresponds to the same
        range reserved in IPP.  Implementers are warned that use of
        such values may conflict with other implementations.
        Implementers are encouraged to request registration of enum
        values following the procedures in Section 3.7.1.

        See Section 3.2 entitled 'The Attribute Mechanism' for a
        description of this textual-convention and its use in the
        jmAttributeTable.  See Section 3.3.8 for the specification of
        each attribute.  The comment(s) after each enum assignment
        specifies the data type(s) of the attribute.

        This is a type 2 enumeration.  See Section 3.7.1.2."

    SYNTAX      INTEGER {
        other(1),                       -- Integer32 (-2..2147483647)
                                        -- AND/OR
                                        -- OCTET STRING(SIZE(0..63))

        -- Job State attributes:
        jobStateReasons2(3),            -- JmJobStateReasons2TC
        jobStateReasons3(4),            -- JmJobStateReasons3TC
        jobStateReasons4(5),            -- JmJobStateReasons4TC
        processingMessage(6),           -- JmUTF8StringTC (SIZE(0..63))
        processingMessageNaturalLangTag(7),
                                        -- OCTET STRING(SIZE(0..63))
        jobCodedCharSet(8),             -- CodedCharSet
        jobNaturalLanguageTag(9),       -- OCTET STRING(SIZE(0..63))
```

```
          -- Job Identification attributes:
          jobURI(20),                     -- OCTET STRING(SIZE(0..63))
          jobAccountName(21),             -- OCTET STRING(SIZE(0..63))
          serverAssignedJobName(22),      -- JmJobStringTC (SIZE(0..63))
          jobName(23),                    -- JmJobStringTC (SIZE(0..63))
          jobServiceTypes(24),            -- JmJobServiceTypesTC
          jobSourceChannelIndex(25),      -- Integer32 (0..2147483647)
          jobSourcePlatformType(26),      -- JmJobSourcePlatformTypeTC
          submittingServerName(27),       -- JmJobStringTC (SIZE(0..63))
          submittingApplicationName(28),  -- JmJobStringTC (SIZE(0..63))
          jobOriginatingHost(29),         -- JmJobStringTC (SIZE(0..63))
          deviceNameRequested(30),        -- JmJobStringTC (SIZE(0..63))
          queueNameRequested(31),         -- JmJobStringTC (SIZE(0..63))
          physicalDevice(32),             -- hrDeviceIndex
                                          -- AND/OR
                                          -- JmUTF8StringTC (SIZE(0..63))
          numberOfDocuments(33),          -- Integer32 (-2..2147483647)
          fileName(34),                   -- JmJobStringTC (SIZE(0..63))
          documentName(35),               -- JmJobStringTC (SIZE(0..63))
          jobComment(36),                 -- JmJobStringTC (SIZE(0..63))
          documentFormatIndex(37),        -- Integer32 (0..2147483647)
          documentFormat(38),             -- PrtInterpreterLangFamilyTC
                                          -- AND/OR
                                          -- OCTET STRING(SIZE(0..63))

          -- Job Parameter attributes:
          jobPriority(50),                -- Integer32 (-2..100)
          jobProcessAfterDateAndTime(51), -- DateAndTime (SNMPv2-TC)
          jobHold(52),                    -- JmBooleanTC
          jobHoldUntil(53),               -- JmJobStringTC (SIZE(0..63))
          outputBin(54),                  -- Integer32 (0..2147483647)
                                          -- AND/OR
                                          -- JmJobStringTC (SIZE(0..63))
          sides(55),                      -- Integer32 (-2..2)
          finishing(56),                  -- JmFinishingTC

          -- Image Quality attributes:
          printQualityRequested(70),      -- JmPrintQualityTC
          printQualityUsed(71),           -- JmPrintQualityTC
          printerResolutionRequested(72), -- JmPrinterResolutionTC
          printerResolutionUsed(73),      -- JmPrinterResolutionTC
          tonerEcomonyRequested(74),      -- JmTonerEconomyTC
          tonerEcomonyUsed(75),           -- JmTonerEconomyTC
          tonerDensityRequested(76),      -- Integer32 (-2..100)
          tonerDensityUsed(77),           -- Integer32 (-2..100)
```

```
        -- Job Progress attributes:
        jobCopiesRequested(90),         -- Integer32 (-2..2147483647)
        jobCopiesCompleted(91),         -- Integer32 (-2..2147483647)
        documentCopiesRequested(92),    -- Integer32 (-2..2147483647)
        documentCopiesCompleted(93),    -- Integer32 (-2..2147483647)
        jobKOctetsTransferred(94),      -- Integer32 (-2..2147483647)
        sheetCompletedCopyNumber(95),   -- Integer32 (-2..2147483647)
        sheetCompletedDocumentNumber(96),
                                        -- Integer32 (-2..2147483647)
        jobCollationType(97),           -- JmJobCollationTypeTC

        -- Impression attributes:
        impressionsSpooled(110),        -- Integer32 (-2..2147483647)
        impressionsSentToDevice(111),   -- Integer32 (-2..2147483647)
        impressionsInterpreted(112),    -- Integer32 (-2..2147483647)
        impressionsCompletedCurrentCopy(113),
                                        -- Integer32 (-2..2147483647)
        fullColorImpressionsCompleted(114),
                                        -- Integer32 (-2..2147483647)
        highlightColorImpressionsCompleted(115),
                                        -- Integer32 (-2..2147483647)

        -- Page attributes:
        pagesRequested(130),            -- Integer32 (-2..2147483647)
        pagesCompleted(131),            -- Integer32 (-2..2147483647)
        pagesCompletedCurrentCopy(132), -- Integer32 (-2..2147483647)

        -- Sheet attributes:
        sheetsRequested(150),           -- Integer32 (-2..2147483647)
        sheetsCompleted(151),           -- Integer32 (-2..2147483647)
        sheetsCompletedCurrentCopy(152),-- Integer32 (-2..2147483647)

        -- Resource attributes:
        mediumRequested(170),           -- JmMediumTypeTC
                                        -- AND/OR
                                        -- JmJobStringTC (SIZE(0..63))
        mediumConsumed(171),            -- Integer32 (-2..2147483647)
                                        -- AND
                                        -- JmJobStringTC (SIZE(0..63))
        colorantRequested(172),         -- Integer32 (-2..2147483647)
                                        -- AND/OR
                                        -- JmJobStringTC (SIZE(0..63))
        colorantConsumed(173),          -- Integer32 (-2..2147483647)
                                        -- AND/OR
                                        -- JmJobStringTC (SIZE(0..63))
        mediumTypeConsumed(174),        -- Integer32 (-2..2147483647)
                                        -- AND
                                        -- JmJobStringTC (SIZE(0..63))
        mediumSizeConsumed(175),        -- Integer32 (-2..2147483647)
                                        -- AND
                                        -- JmJobStringTC (SIZE(0..63))
        -- Time attributes:
        jobSubmissionToServerTime(190), -- JmTimeStampTC
                                        -- AND/OR
                                        -- DateAndTime
        jobSubmissionTime(191),         -- JmTimeStampTC
                                        -- AND/OR
                                        -- DateAndTime
        jobStartedBeingHeldTime(192),   -- JmTimeStampTC
                                        -- AND/OR
                                        -- DateAndTime
        jobStartedProcessingTime(193),  -- JmTimeStampTC
                                        -- AND/OR
                                        -- DateAndTime
        jobCompletionTime(194),         -- JmTimeStampTC
                                        -- AND/OR
                                        -- DateAndTime
        jobProcessingCPUTime(195)       -- Integer32 (-2..2147483647)
    }
```

```
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 jobServiceTypes
        attribute, corresponding to the hexadecimal values: 0x8 + 0x20
        + 0x4, respectively, yielding: 0x2C.

        Whether this attribute is set from a job attribute supplied by
        the job submission client or is set by the recipient job
        submission server or device depends on the job submission
        protocol.  With either implementation, the agent SHALL return a
        non-zero value for this attribute indicating the type of the
        job.

        One of the purposes of this attribute is to permit a requester
        to filter out jobs that are not of interest.  For example, a
        printer operator MAY only be interested in jobs that include
        printing.  That is why the attribute is in the job
        identification category.

        The following service component types are defined (in
        hexadecimal) and are assigned a separate bit value for use with
        the jobServiceTypes attribute:

        other                           0x1
            The job contains some instructions that are not one of the
            identified types.

        unknown                         0x2
            The job contains some instructions whose type is unknown to
            the agent.

        print                           0x4
            The job contains some instructions that specify printing

        scan                            0x8
            The job contains some instructions that specify scanning
        faxIn                             0x10
            The job contains some instructions that specify receive fax

        faxOut                            0x20
            The job contains some instructions that specify sending fax

        getFile                           0x40
            The job contains some instructions that specify accessing
            files or documents

        putFile                           0x80
            The job contains some instructions that specify storing
            files or documents

        mailList                          0x100
            The job contains some instructions that specify
            distribution of documents using an electronic mail system.

        These bit definitions are the equivalent of a type 2 enum
        except that combinations of them MAY be used together.  See
        section 3.7.1.2."
    SYNTAX      INTEGER (0..2147483647)   -- 31 bits, all but sign bit



JmJobStateReasons1TC ::= TEXTUAL-CONVENTION
    STATUS      current
    DESCRIPTION
        "The JmJobStateReasonsNTC (N=1..4) textual-conventions are used
        with the jmJobStateReasons1 object and jobStateReasonsN
        (N=2..4), respectively, to provide additional information
        regarding the current jmJobState object value.  These values
        MAY be used with any job state or states for which the reason
        makes sense.  See section 3.3.9.1 for the specification of each
        bit value defined for use with the JmJobStateReasons1TC.

        These bit definitions are the equivalent of a type 2 enum
        except that combinations of bits may be used together.  See
        section 3.7.1.2."
    SYNTAX      INTEGER (0..2147483647)   -- 31 bits, all but sign bit



JmJobStateReasons2TC ::= TEXTUAL-CONVENTION
    STATUS      current
    DESCRIPTION
        "This textual-convention is used with the jobStateReasons2
        attribute to provides additional information regarding the
        jmJobState object.  See section 3.3.9.2 for the specification
        of JmJobStateReasons2TC.  See section 3.3.9.1 for the
        description under JmJobStateReasons1TC for additional
        information that applies to all reasons.
```

```

            These bit definitions are the equivalent of a type 2 enum
            except that combinations of them may be used together.  See
            section 3.7.1.2."
        SYNTAX      INTEGER (0..2147483647)   -- 31 bits, all but sign bit

    JmJobStateReasons3TC ::= TEXTUAL-CONVENTION
        STATUS      current
        DESCRIPTION
            "This textual-convention is used with the jobStateReasons3
            attribute to provides additional information regarding the
            jmJobState object.  See section 3.3.9.3 for the specification
            of JmJobStateReasons3TC.  See section 3.3.9.1 for the
            description under JmJobStateReasons1TC for additional
            information that applies to all reasons.

            These bit definitions are the equivalent of a type 2 enum
            except that combinations of them may be used together.  See
            section 3.7.1.2.  "
        SYNTAX      INTEGER (0..2147483647)   -- 31 bits, all but sign bit





    JmJobStateReasons4TC ::= TEXTUAL-CONVENTION
        STATUS      current
        DESCRIPTION
            "This textual-convention is used in the jobStateReasons4
            attribute to provides additional information regarding the
            jmJobState object.  See section 3.3.9.4 for the specification
            of JmJobStateReasons4TC.  See section 3.3.9.1 for the
            description under JmJobStateReasons1TC for additional
            information that applies to all reasons.

            These bit definitions are the equivalent of a type 2 enum
            except that combinations of them may be used together.  See
            section 3.7.1.2."
        SYNTAX      INTEGER (0..2147483647)   -- 31 bits, all but sign bit
```

```
jobmonMIBObjects  OBJECT IDENTIFIER  ::= { jobmonMIB 1 }

-- The General Group (MANDATORY)

-- The jmGeneralGroup consists entirely of the jmGeneralTable.

jmGeneral  OBJECT IDENTIFIER ::= { jobmonMIBObjects 1 }

jmGeneralTable  OBJECT-TYPE
    SYNTAX      SEQUENCE OF JmGeneralEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The jmGeneralTable consists of information of a general nature
        that are per-job-set, but are not per-job.  See Section 2
        entitled 'Terminology and Job Model' for the definition of a
        job set.

        The MANDATORY-GROUP macro specifies that this group is
        MANDATORY."
    ::= { jmGeneral 1 }


jmGeneralEntry  OBJECT-TYPE
    SYNTAX      JmGeneralEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "Information about a job set (queue).

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


JmGeneralEntry ::= SEQUENCE {
    jmGeneralJobSetIndex                Integer32 (1..32767),
    jmGeneralNumberOfActiveJobs         Integer32 (0..2147483647),
    jmGeneralOldestActiveJobIndex       Integer32 (0..2147483647),
    jmGeneralNewestActiveJobIndex       Integer32 (0..2147483647),
    jmGeneralJobPersistence             Integer32 (15..2147483647),
    jmGeneralAttributePersistence       Integer32 (15..2147483647),
    jmGeneralJobSetName                 JmUTF8StringTC (SIZE(0..63))
}
```

```
jmGeneralJobSetIndex OBJECT-TYPE
    SYNTAX      Integer32 (1..32767)
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A unique value for each job set in this MIB.  The jmJobTable
        and jmAttributeTable tables have this same index as their
        primary index.

        The value(s) of the jmGeneralJobSetIndex SHALL be persistent
        across power cycles, so that clients that have retained
        jmGeneralJobSetIndex values will access the same job sets upon
        subsequent power-up.

        An implementation that has only one job set, such as a printer
        with a single queue, SHALL hard code this object with the value
        1.

        See Section 2 entitled 'Terminology and Job Model' for the
        definition of a job set.
        Corresponds to the first index in jmJobTable and
        jmAttributeTable."
    ::= { jmGeneralEntry 1 }


jmGeneralNumberOfActiveJobs OBJECT-TYPE
    SYNTAX      Integer32 (0..2147483647)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The current number of 'active' jobs in the jmJobIDTable,
        jmJobTable, and jmAttributeTable, i.e., the total number of
        jobs that are in the pending, processing, or processingStopped
        states.  See the JmJobStateTC textual-convention for the exact
        specification of the semantics of the job states."
    DEFVAL      { 0 }      -- no jobs
    ::= { jmGeneralEntry 2 }
```

```
jmGeneralOldestActiveJobIndex  OBJECT-TYPE
    SYNTAX      Integer32 (0..2147483647)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The jmJobIndex of the oldest job that is still in one of the
        'active' states (pending, processing, or processingStopped).
        In other words, the index of the 'active' job that has been in
        the job tables the longest.

        If there are no active jobs, the agent SHALL set the value of
        this object to 0.

        See Section 3.2 entitled 'The Job Tables and the Oldest Active
        and Newest Active Indexes' for a description of the usage of
        this object."
    DEFVAL      { 0 }      -- no active jobs
    ::= { jmGeneralEntry 3 }



jmGeneralNewestActiveJobIndex  OBJECT-TYPE
    SYNTAX      Integer32 (0..2147483647)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The jmJobIndex of the newest job that is in one of the
        'active' states (pending, processing, or processingStopped).
        In other words, the index of the 'active' job that has been
        most recently added to the job tables.

        When all jobs become 'inactive', i.e., enter the pendingHeld,
        completed, canceled, or aborted states, the agent SHALL set the
        value of this object to 0.

        See Section 3.2 entitled 'The Job Tables and the Oldest Active
        and Newest Active Indexes' for a description of the usage of
        this object."
    DEFVAL      { 0 }      -- no active jobs
    ::= { jmGeneralEntry 4 }
```

```
jmGeneralJobPersistence OBJECT-TYPE
    SYNTAX      Integer32 (15..2147483647)
    UNITS       "seconds"
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The minimum time in seconds for this instance of the Job Set
        that an entry SHALL remain in the jmJobIDTable and jmJobTable
        after processing has completed, i.e., the minimum time in
        seconds starting when the job enters the completed, canceled,
        or aborted state.

        Configuring this object is implementation-dependent.

        This value SHALL be equal to or greater than the value of
        jmGeneralAttributePersistence.  This value SHOULD be at least
        60 which gives a monitoring or accounting application one
        minute in which to poll for job data."
    DEFVAL      { 60 }          -- one minute
    ::= { jmGeneralEntry 5 }



jmGeneralAttributePersistence OBJECT-TYPE
    SYNTAX      Integer32 (15..2147483647)
    UNITS       "seconds"
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The minimum time in seconds for this instance of the Job Set
        that an entry SHALL remain in the jmAttributeTable after
        processing has completed , i.e., the time in seconds starting
        when the job enters the completed, canceled, or aborted state.

        Configuring this object is implementation-dependent.

        This value SHOULD be at least 60 which gives a monitoring or
        accounting application one minute in which to poll for job
        data."
    DEFVAL      { 60 }          -- one minute
    ::= { jmGeneralEntry 6 }
```

```
jmGeneralJobSetName OBJECT-TYPE
    SYNTAX      JmUTF8StringTC (SIZE(0..63))
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The human readable name of this job set assigned by the system
        administrator (by means outside of this MIB).  Typically, this
        name SHOULD be the name of the job queue.  If a server or
        device has only a single job set, this object can be the
        administratively assigned name of the server or device itself.
        This name does not need to be unique, though each job set in a
        single Job Monitoring MIB SHOULD have distinct names.

        NOTE - If the job set corresponds to a single printer and the
        Printer MIB is implemented, this value SHOULD be the same as
        the prtGeneralPrinterName object in the draft Printer MIB
        [print-mib-draft].  If the job set corresponds to an IPP
        Printer, this value SHOULD be the same as the IPP 'printer-
        name' Printer attribute.

        NOTE - 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.

        See the OBJECT compliance macro for the minimum maximum length
        required for conformance."
    DEFVAL      { ''H }      -- empty string
    ::= { jmGeneralEntry 7 }
```

```
-- The Job ID Group (MANDATORY)

-- The jmJobIDGroup consists entirely of the jmJobIDTable.

jmJobID  OBJECT IDENTIFIER ::= { jobmonMIBObjects 2 }

jmJobIDTable  OBJECT-TYPE
    SYNTAX      SEQUENCE OF JmJobIDEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The jmJobIDTable provides a correspondence map (1) between the
        job submission ID that a client uses to refer to a job and (2)
        the jmGeneralJobSetIndex and jmJobIndex that the Job Monitoring
        MIB agent assigned to the job and that are used to access the
        job in all of the other tables in the MIB.  If a monitoring
        application already knows the jmGeneralJobSetIndex and the
        jmJobIndex of the job it is querying, that application NEED NOT
        use the jmJobIDTable.

        The MANDATORY-GROUP macro specifies that this group is
        MANDATORY."
    ::= { jmJobID 1 }



jmJobIDEntry  OBJECT-TYPE
    SYNTAX      JmJobIDEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The map from (1) the jmJobSubmissionID to (2) the
        jmGeneralJobSetIndex and jmJobIndex.

        An entry SHALL exist in this table for each job currently known
        to the agent for all job sets and job states.  There MAY be
        more than one jmJobIDEntry that maps to a single job.  This
        many to one mapping can occur when more than one network entity
        along the job submission path supplies a job submission ID.
        See Section 3.5.  However, each job SHALL appear once and in
        one and only one job set."
    INDEX  { jmJobSubmissionID }
    ::= { jmJobIDTable 1 }

JmJobIDEntry ::= SEQUENCE {
    jmJobSubmissionID                      OCTET STRING(SIZE(48)),
    jmJobIDJobSetIndex                     Integer32 (0..32767),
    jmJobIDJobIndex                        Integer32 (0..2147483647)
}
```

```
jmJobSubmissionID OBJECT-TYPE
    SYNTAX      OCTET STRING(SIZE(48))
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A quasi-unique 48-octet fixed-length string ID which
        identifies the job within a particular client-server
        environment.  There are multiple formats for the
        jmJobSubmissionID.  Each format SHALL be uniquely identified.
        See the JmJobSubmissionIDTypeTC textual convention.  Each
        format SHALL be registered using the procedures of a type 2
        enum.  See section 3.7.3 entitled: 'PWG Registration of Job
        Submission Id Formats'.

        If the requester (client or server) does not supply a job
        submission ID in the job submission protocol, then the
        recipient (server or device) SHALL assign a job submission ID
        using any of the standard formats that have been reserved for
        agents and adding the final 8 octets to distinguish the ID from
        others submitted from the same requester.

        The monitoring application, whether in the client or running
        separately, MAY use the job submission ID to help identify
        which jmJobIndex was assigned by the agent, i.e., in which row
        the job information is in the other tables.

        NOTE - fixed-length is used so that a management application
        can use a shortened GetNext varbind (in SNMPv1 and SNMPv2) in
        order to get the next submission ID, disregarding the remainder
        of the ID in order to access jobs independent of the trailing
        identifier part, e.g., to get all jobs submitted by a
        particular jmJobOwner or submitted from a particular MAC
        address.

        See the JmJobSubmissionIDTypeTC textual convention.
        See APPENDIX B - Support of Job Submission Protocols."
    ::= { jmJobIDEntry 1 }
```

```
jmJobIDJobSetIndex OBJECT-TYPE
    SYNTAX      Integer32 (0..32767)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "This object contains the value of the jmGeneralJobSetIndex for
        the job with the jmJobSubmissionID value, i.e., the job set
        index of the job set in which the job was placed when that
        server or device accepted the job.  This 16-bit value in
        combination with the jmJobIDJobIndex value permits the
        management application to access the other tables to obtain the
        job-specific objects for this job.

        See jmGeneralJobSetIndex in the jmGeneralTable."
    DEFVAL      { 0 }      -- 0 indicates no job set index
    ::= { jmJobIDEntry 2 }



jmJobIDJobIndex OBJECT-TYPE
    SYNTAX      Integer32 (0..2147483647)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "This object contains the value of the jmJobIndex for the job
        with the jmJobSubmissionID value, i.e., the job index for the
        job when the server or device accepted the job.  This value, in
        combination with the jmJobIDJobSetIndex value, permits the
        management application to access the other tables to obtain the
        job-specific objects for this job.

        See jmJobIndex in the jmJobTable."
    DEFVAL      { 0 }      -- 0 indicates no jmJobIndex value.
    ::= { jmJobIDEntry 3 }
```

```
-- The Job Group (MANDATORY)

-- The jmJobGroup consists entirely of the jmJobTable.

jmJob  OBJECT IDENTIFIER ::= { jobmonMIBObjects 3 }

jmJobTable  OBJECT-TYPE
    SYNTAX      SEQUENCE OF JmJobEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The jmJobTable consists of basic job state 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 MANDATORY-GROUP macro specifies that this group is
        MANDATORY."
    ::= { jmJob 1 }



jmJobEntry  OBJECT-TYPE
    SYNTAX      JmJobEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "Basic per-job state 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.

        See Section 3.2 entitled 'The Job Tables'."
    INDEX  { jmGeneralJobSetIndex, jmJobIndex }
    ::= { jmJobTable 1 }

JmJobEntry ::= SEQUENCE {
    jmJobIndex                          Integer32 (1..2147483647),
    jmJobState                          JmJobStateTC,
    jmJobStateReasons1                  JmJobStateReasons1TC,
    jmNumberOfInterveningJobs           Integer32 (-2..2147483647),
    jmJobKOctetsPerCopyRequested        Integer32 (-2..2147483647),
    jmJobKOctetsProcessed               Integer32 (-2..2147483647),
    jmJobImpressionsPerCopyRequested    Integer32 (-2..2147483647),
    jmJobImpressionsCompleted           Integer32 (-2..2147483647),
    jmJobOwner                          JmJobStringTC (SIZE(0..63))
}
```

```
jmJobIndex OBJECT-TYPE
    SYNTAX      Integer32 (1..2147483647)
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The sequential, monatonically increasing identifier index for
        the job generated by the server or device when that server or
        device accepted the job.  This index value permits the
        management application to access the other tables to obtain the
        job-specific row entries.

        See Section 3.2 entitled 'The Job Tables and the Oldest Active
        and Newest Active Indexes'.
        See Section 3.5 entitled 'Job Identification'.
        See also jmGeneralNewestActiveJobIndex for the largest value of
        jmJobIndex.
        See JmJobSubmissionIDTypeTC for a limit on the size of this
        index if the agent represents it as an 8-digit decimal number."
    ::= { jmJobEntry 1 }



jmJobState OBJECT-TYPE
    SYNTAX      JmJobStateTC
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The current state of the job (pending, processing, completed,
        etc.).  Agents SHALL implement only those states which are
        appropriate for the particular implementation.  However,
        management applications SHALL be prepared to receive all the
        standard job states.

        The final value for this object SHALL be one of: completed,
        canceled, or aborted.  The minimum length of time that the
        agent SHALL maintain MIB data for a job in the completed,
        canceled, or aborted state before removing the job data from
        the jmJobIDTable and jmJobTable is specified by the value of
        the jmGeneralJobPersistence object."
    DEFVAL      { unknown }      -- default is unknown
    ::= { jmJobEntry 2 }
```

```
jmJobStateReasons1 OBJECT-TYPE
    SYNTAX      JmJobStateReasons1TC
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "Additional information about the job's current state, i.e.,
        information that augments the value of the job's jmJobState
        object.

        Implementation of any reason values is OPTIONAL, but an agent
        SHOULD return any reason information available.  These values
        MAY be used with any job state or states for which the reason
        makes sense.  Since the Job State Reasons will be more dynamic
        than the Job State, it is recommended that a job monitoring
        application read this object every time jmJobState is read.
        When the agent cannot provide a reason for the current state of
        the job, the value of the jmJobStateReasons1 object and
        jobStateReasonsN attributes SHALL be 0.

        The jobStateReasonsN (N=2..4) attributes provide further
        additional information about the job's current state."
    DEFVAL      { 0 }      -- no reasons
    ::= { jmJobEntry 3 }



jmNumberOfInterveningJobs OBJECT-TYPE
    SYNTAX      Integer32 (-2..2147483647)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The number of jobs that are expected to complete processing
        before this job has completed processing according to the
        implementation's queuing algorithm, if no other jobs were to be
        submitted.  In other words, this value is the job's queue
        position.  The agent SHALL return a value of 0 for this
        attribute when the job is the next job to complete processing
        (or has completed processing)."
    DEFVAL      { 0 }      -- default is no intervening jobs.
    ::= { jmJobEntry 4 }
```

```
jmJobKOctetsPerCopyRequested OBJECT-TYPE
    SYNTAX      Integer32 (-2..2147483647)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The total size in K (1024) octets of the document(s) being
        requested to be processed in the job.  The agent SHALL round
        the actual number of octets up to the next highest K.  Thus 0
        octets is represented as '0', 1-1024 octets is represented as
        '1', 1025-2048 is represented as '2', etc.

        In computing this value, the server/device SHALL NOT 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 indicates the size of the document(s)
        measured in K octets independent of the number of copies."
    DEFVAL      { -2 }      -- the default is unknown(-2)
    ::= { jmJobEntry 5 }



jmJobKOctetsProcessed OBJECT-TYPE
    SYNTAX      Integer32 (-2..2147483647)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The total number of octets processed by the server or device
        measured in units of K (1024) octets so far.  The agent SHALL
        round the actual number of octets processed up to the next
        higher K.  Thus 0 octets is represented as '0', 1-1024 octets
        is represented as '1', 1025-2048 octets is '2', etc.  For
        printing devices, this value is the number interpreted by the
        page description language interpreter rather than what has been
        marked on media.

        For implementations where multiple copies are produced by the
        interpreter with only a single pass over the data, the final
        value SHALL be equal to the value of the
        jmJobKOctetsPerCopyRequested object.  For implementations where
        multiple copies are produced by the interpreter by processing
        the data for each copy, the final value SHALL be a multiple of
        the value of the jmJobKOctetsPerCopyRequested object.

        NOTE - See the impressionsCompletedCurrentCopy and
        pagesCompletedCurrentCopy attributes for attributes that are
        reset on each document copy.
        NOTE - The jmJobKOctetsProcessed object can be used with the
        jmJobKOctetsPerCopyRequested object to provide an indication of
        the relative progress of the job, provided that the
        multiplicative factor is taken into account for some
        implementations of multiple copies."
    DEFVAL      { 0 }       -- default is no octets processed.
    ::= { jmJobEntry 6 }


jmJobImpressionsPerCopyRequested OBJECT-TYPE
    SYNTAX      Integer32 (-2..2147483647)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The total size in number of impressions of the document(s)
        submitted.

        In computing this value, the server/device SHALL NOT 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 reflects the size of the document(s)
        measured in impressions independent of the number of copies.

        See the definition of the term 'impression' in Section 2."
    DEFVAL      { -2 }      -- default is unknown(-2)
    ::= { jmJobEntry 7 }


jmJobImpressionsCompleted OBJECT-TYPE
    SYNTAX      Integer32 (-2..2147483647)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The total number of impressions completed for this job so far.
        For printing devices, the impressions completed includes
        interpreting, marking, and stacking the output.  For other
        types of job services, the number of impressions completed
        includes the number of impressions processed.

        NOTE - See the impressionsCompletedCurrentCopy and
        pagesCompletedCurrentCopy attributes for attributes that are
        reset on each document copy.

        NOTE - The jmJobImpressionsCompleted object can be used with
        the jmJobImpressionsPerCopyRequested object to provide an
        indication of the relative progress of the job, provided that
        the multiplicative factor is taken into account for some
        implementations of multiple copies.
```

```
            See the definition of the term 'impression' in Section 2 and
            the counting example in Section 3.4 entitled 'Monitoring Job
            Progress'."
        DEFVAL      { 0 }      -- default is no octets
        ::= { jmJobEntry 8 }



    jmJobOwner OBJECT-TYPE
        SYNTAX      JmJobStringTC (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 ensure that the name
            is unique to the network that is visible to the client and
            target device.

            This value SHOULD be the most authenticated name of the user
            submitting the job.

            See the OBJECT compliance macro for the minimum maximum length
            required for conformance."
        DEFVAL      { ''H }      -- default is empty string
        ::= { jmJobEntry 9 }
```

```
-- The Attribute Group (MANDATORY)

-- The jmAttributeGroup consists entirely of the jmAttributeTable.
--
-- Implementation of the objects in this group is MANDATORY.
-- See Section 3.1 entitled 'Conformance Considerations'.
-- An agent SHALL implement any attribute if (1) the server or device
-- supports the functionality represented by the attribute and (2) the
-- information is available to the agent.

jmAttribute  OBJECT IDENTIFIER ::= { jobmonMIBObjects 4 }



jmAttributeTable  OBJECT-TYPE
    SYNTAX      SEQUENCE OF JmAttributeEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The jmAttributeTable SHALL contain attributes of the job and
        document(s) for each job in a job set.  Instead of allocating
        distinct objects for each attribute, each attribute is
        represented as a separate row in the jmAttributeTable.

        The MANDATORY-GROUP macro specifies that this group is
        MANDATORY.  An agent SHALL implement any attribute if (1) the
        server or device supports the functionality represented by the
        attribute and (2) the information is available to the agent. "
    ::= { 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.

        Each entry in the jmAttributeTable is a per-job entry 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
        (see the JmAttributeTypeTC textual-convention).  The value of
        the attribute SHALL be represented in either the
        jmAttributeValueAsInteger or jmAttributeValueAsOctets objects,
        and/or both, as specified in the JmAttributeTypeTC textual-
        convention.

        The agent SHALL create rows in the jmAttributeTable as the
        server or device 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 the agent adds additional rows to
        this table.  As the attributes that represent resources are
        actually consumed, the usage counter contained in the
        jmAttributeValueAsInteger object is incremented according to
        the units indicated in the description of the JmAttributeTypeTC
        enum.

        The agent SHALL maintain each row in the jmAttributeTable for
        at least the minimum time after a job completes as specified by
        the jmGeneralAttributePersistence object.

        Zero or more entries SHALL exist in this table for each job in
        a job set.

        See Section 3.3 entitled 'The Attribute Mechanism' for a
        description of the jmAttributeTable."
    INDEX  { jmGeneralJobSetIndex, jmJobIndex, jmAttributeTypeIndex,
    jmAttributeInstanceIndex }
    ::= { jmAttributeTable 1 }

JmAttributeEntry ::= SEQUENCE {
    jmAttributeTypeIndex                  JmAttributeTypeTC,
    jmAttributeInstanceIndex              Integer32 (1..32767),
    jmAttributeValueAsInteger             Integer32 (-2..2147483647),
    jmAttributeValueAsOctets              OCTET STRING(SIZE(0..63))
}
```

```
jmAttributeTypeIndex OBJECT-TYPE
    SYNTAX      JmAttributeTypeTC
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The type of attribute that this row entry represents.

        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 attributes (i.e., apply to the job as a whole)
        that have only one instance per job include:
        jobCopiesRequested(90), documentCopiesRequested(92),
        jobCopiesCompleted(91), documentCopiesCompleted(93), while
        examples of job attributes that may have more than one instance
        per job include:  documentFormatIndex(37), and
        documentFormat(38).

        Examples of document attributes (one instance per document)
        include: fileName(34), and documentName(35).

        Examples of required and consumed resource attributes include:
        pagesRequested(130), mediumRequested(170), pagesCompleted(131),
        and mediumConsumed(171), respectively."
    ::= { 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 or per document, such as documentFormatIndex(37) or
        documentFormat(38), the index SHALL be a running index for the
        job as a whole, starting at 1."
    ::= { jmAttributeEntry 2 }
```

```
jmAttributeValueAsInteger OBJECT-TYPE
    SYNTAX      Integer32 (-2..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 in the JmAttributeTypeTC textual-convention
        definition 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
        textual-convention, 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(34), jobName, and
        processingMessage, do not have the 'INTEGER:' tag in the
        JmAttributeTypeTC definition and so an agent SHALL always
        return a value of '-1' to indicate 'other' for the value of the
        jmAttributeValueAsInteger object for these attributes.

        For attributes which do have the 'INTEGER:' tag in the
        JmAttributeTypeTC definition, if the integer value is not (yet)
        known, the agent either (1) SHALL not materialize the row in
        the jmAttributeTable until the value is known or (2) SHALL
        return a '-2' to represent an 'unknown' counting integer value,
        a '0' to represent an 'unknown' index value, and a '2' to
        represent an 'unknown(2)' enum value."
    DEFVAL      { -2 }      -- default value is unknown(-2)
    ::= { 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 in the JmAttributeTypeTC textual-convention
        definition has the tag: 'OCTETS:'.

        Depending on the enum definition, this object value MAY be a
        coded character set string (text), such as 'JmUTF8StringTC', 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 the agent
        SHALL always return a zero length string for the value of the
        jmAttributeValueAsOctets object.

        For attributes which do have the 'OCTETS:' tag in the
        JmAttributeTypeTC definition, if the OCTET STRING value is not
        (yet) known, the agent either SHALL NOT materialize the row in
        the jmAttributeTable until the value is known or SHALL return a
        zero-length string."
    DEFVAL      { ''H }      -- empty string
    ::= { jmAttributeEntry 4 }
```

```
-- The Mirror Attribute Group (OPTIONAL)

-- The jmMirrorAttrGroup consists entirely of the jmMirrorAttrTable.
--
-- Implementation of the objects in this group is OPTIONAL.
-- See Section 3.1 entitled 'Conformance Considerations'.
-- The jmMirrorAttrTable complements the MANDATORY jmAttributeTable.
--
-- The jmMirrorAttrTable provides access to all of the attributes that
-- an implementation supports, sorted by attribute type (traditional
-- SNMP MIB access), rather than being sorted by job set and job index
-- (modern object-oriented access) as in the analogous
-- jmAttributeTable.

jmMirrorAttr    OBJECT IDENTIFIER ::= { jobmonMIBObjects 5 }

jmMirrorAttrTable  OBJECT-TYPE
    SYNTAX      SEQUENCE OF JmMirrorAttrEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The jmMirrorAttrTable is an OPTIONAL table which provides
        identical attributes to the jmAttributeTable but with a
        different index structure.  See jmAttributeTable for further
        details.

        See Section 3.3 entitled 'The Attribute Mechanism' for a
        description of the jmMirrorAttrTable."
    ::= { jmMirrorAttr 1 }
```

```
jmMirrorAttrEntry  OBJECT-TYPE
    SYNTAX      JmMirrorAttrEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The attributes that represent information about each job and
        documents or resources required and/or consumed.

        Each entry in jmMirrorAttrTable is a per-attribute entry with a
        primary index for each type of attribute jmMirrorAttrTypeIndex)
        that a job can have and secondary indices which specify job set
        (jmJobSetIndex), job instance (jmJobIndex), and attribute
        instance (jmMirrorAttrInstanceIndex).

        An agent which implements the jmMirrorAttrTable SHALL create
        and maintain a row in the jmMirrorAttrTable for each
        corresponding row in the jmAttributeTable."
    INDEX  { jmMirrorAttrTypeIndex, jmGeneralJobSetIndex, jmJobIndex,
    jmMirrorAttrInstanceIndex }
    ::= { jmMirrorAttrTable 1 }

JmMirrorAttrEntry ::= SEQUENCE {
    jmMirrorAttrTypeIndex                JmAttributeTypeTC,
    jmMirrorAttrInstanceIndex            Integer32 (1..32767),
    jmMirrorAttrValueAsInteger           Integer32 (-2..2147483647),
    jmMirrorAttrValueAsOctets            OCTET STRING(SIZE(0..63))
}

jmMirrorAttrTypeIndex OBJECT-TYPE
    SYNTAX      JmAttributeTypeTC
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The type of attribute that this row entry represents.

        See jmAttributeTypeIndex in jmAttributeTable for complete
        description."
    ::= { jmMirrorAttrEntry 1 }

jmMirrorAttrInstanceIndex OBJECT-TYPE
    SYNTAX      Integer32 (1..32767)
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The instance of attribute that this row entry represents.

        See jmAttributeInstanceIndex in jmAttributeTable for complete
        description."
    ::= { jmMirrorAttrEntry 2 }
```

```
jmMirrorAttrValueAsInteger OBJECT-TYPE
    SYNTAX      Integer32 (-2..2147483647)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The integer value of the attribute.

        See jmAttributeValueAsInteger in jmAttributeTable for complete
        description."
    DEFVAL      { -2 }      -- default value is unknown(-2)
    ::= { jmMirrorAttrEntry 3 }

jmMirrorAttrValueAsOctets OBJECT-TYPE
    SYNTAX      OCTET STRING(SIZE(0..63))
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The octet string value of the attribute.

        See jmAttributeValueAsOctets in jmAttributeTable for complete
        description."
    DEFVAL      { ’’H }      -- empty string
    ::= { jmMirrorAttrEntry 4 }
```

```


-- The System Group (MANDATORY)
-- (This group was added in version 1.3 of this MIB).

-- The jmMirrorAttrGroup consists entirely of objects that summarize
-- the implementation of this MIB on a system.

jmSystem        OBJECT IDENTIFIER ::= { jobmonMIBObjects 6 }

jmSystemVersionString OBJECT-TYPE
    SYNTAX      JmUTF8StringTC
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The minor and minor version of this MIB implemented by this
        system.

        The format of the string SHALL be the ASCII major version
        number followed by an ASCII PERIOD (.), followed by the ASCII
        minor version number, i.e., '1.3' for this version."
    DEFVAL  { '312E33'H }                -- version 1.3
    ::= { jmSystem 1 }

jmSystemOptionSupport OBJECT-TYPE
    SYNTAX      INTEGER(0..2147483647)    -- biggest int 2**31 - 1
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The options of the MIB specification that this implementation
        supports specified as a bit mask.

        The current set of values (which may be extended in the future)
        is given below:

               1 : jmMirrorAttrGroup                -- 2**0   OPTIONAL

        Example:  An implementation supporting the jmMirrorAttrGroup
        would return an integer value of { 1 }.

        This object helps a management application determine which MIB
        options are supported in this system."
    DEFVAL    { 0 }                       -- no options are required
    ::= { jmSystem 2 }

```

```
jmSystemAttrIntegerSupport OBJECT-TYPE
    SYNTAX      OCTET STRING (SIZE (0..63))
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "A bit array indicating which attributes of the MIB this
        implementation supports with meaningful integer values.

        The value of this object is a sparse bit array in which bit n
        is a 1 if attribute n is supported with the
        jmAttributeValueAsInteger object with meaningful values, where
        n is the value of the enumerated attribute type in the
        JmAttributeTypeTC used in jmAttributeTypeIndex (and the
        jmMirrorAttrTypeIndex if the jmMirrorAttrTable is implemented).
        Bit n MUST be 0 (or beyond the end of the returned bit array),
        if attribute n is not supported or is always returned with a '-
        1'(other) or '-2'(unknown) value.

        The high order bit of the first octet in this octet string
        corresponds to an attribute type of 0 (reserved), i.e., the bit
        string uses the Big Endian numbering convention.  Compare with
        the BITS data type in SMIv2 [SMIv2-SMI] which has the same
        format but requires contiguous enumerated bits.  Trailing
        octets in the octet string that contain only zero bits MUST NOT
        be returned.

        Note:  private attributes cannot be represented in this bit
        array because their enum values are in the range 2**30 to
        2**31-1.  See section 3.3.8.

        Example:  An implementation supporting the attributes:
        jobStateReasons2(3), jobStateReasons3(4), and jobName(23)
        would return a one-octet string value of { '18'H }, since
        jobName is an octet string value, not an integer value.

        This object helps a management application determine which
        attributes with meaningful integer values MAY be present on
        jobs in this system."
    DEFVAL   { ''H }                    -- no attributes are required
    ::= { jmSystem 3 }
```

```
jmSystemAttrOctetsSupport OBJECT-TYPE
    SYNTAX      OCTET STRING (SIZE (0..63))
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "A bit array indicating which attributes of the MIB this
        implementation supports with meaningful octet string values.

        The format and semantics of this object is the same as
        jmSystemAttrIntegerSupport, except that bit n indicates that
        attribute n supports the jmAttributeValueAsOctets object with
        meaningful values, instead of the jmAttributeValueAsInteger
        object.  Bit n MUST be 0 (or beyond the end of the returned bit
        array), if attribute n is not supported or is always returned
        as a zero-length octet string value.

        If an implementation supports both jmAttributeValueAsInteger
        and jmAttributeValueAsOctets with meaningful values for
        attribute n, bit n MUST appear in both bit array objects with a
        1 value.

        Example:  An implementation supporting the attributes:
        jobStateReasons2(3), jobStateReasons3(4), and jobName(23)
        would return a three-octet string value of { '000001'H }, since
        jobStateReasons2 and jobStateReasons3 are integer values, not
        octet string values.

        This object helps a management application determine which
        attributes with meaningful octet string values MAY be present
        on jobs in this system."
    DEFVAL   { ''H }                      -- no attributes are required
    ::= { jmSystem 4 }
```

```
jmSystemAttrMultiRowSupport OBJECT-TYPE
    SYNTAX      OCTET STRING (SIZE (0..63))
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "A bit array indicating which MULTI-ROW attributes of the MIB
        this implementation supports with multiple integer values
        and/or multiple octet string values.

        The format of this object is the same as the
        jmSystemAttrIntegerSupport and jmSystemAttrOctetsSupport
        objects.  Bit n MUST be 1, if attribute n is actually supported
        with more than one integer and/or more than one octet string
        value.  Bit n MUST be 0 (or beyond the end of the returned bit
        array), if attribute n is not supported, is always returned as
        a single integer value, or as a single octet string value.  For
        every bit n that is a 1 in this bit array, there MUST be a
        corresponding 1 for bit n in either jmSystemAttrIntegerSupport,
        jmSystemAttrOctetsSupport, or both.

        Example:  Consider an implementation supporting:
        (a) the jobStateReasons2(3), jobStateReasons3(4) SINGLE-ROW
        integer attributes
        (b) the jobName(23) SINGLE-ROW octet string attribute
        (c) more than one integer value for the mediumRequested(170)
        and mediumConsumed(171) MULTI-ROW attributes AND
        (d) more than one octet string value for the fileName(34),
        documentName(35), and mediumConsumed(171) MULTI-ROW attributes
        (e) no octet string values for mediumRequested(170).
        Such an implementation would return:
        jmSystemAttrIntegerSupport 22 octets:
          { '18000000 00000000 00000000 00000000 00000000 0030'H }
        jmSystemAttrOctetsSupport 22 octets:
          { '00000100 30000000 00000000 00000000 00000000 0010'H }
        jmSystemAttrMultiRowSupport 22 octets:
          { '00000000 30000000 00000000 00000000 00000000 0030'H }

        Example:  Consider an implementation that supports the
        fileName(34) MULTI-ROW attribute, but does not support more
        than one document per job.  Such an implementation would NOT
        return a 1 bit for bit 34 in jmSystemAttrMultiRowSupport, since
        such an implementation would never return more than one
        fileName value for a job.  It would return a zero-length
        string, since it never returns more than one value.

        This object helps a management application determine which
        attributes may return more than one integer value or more than
        one octet string value on jobs in this system."
    DEFVAL   { ''H }                      -- no attributes are required
    ::= { jmSystem 5 }
```

```
-- Notifications and Trapping
-- Reserved for the future

jobmonMIBNotifications  OBJECT IDENTIFIER  ::= { jobmonMIB 2 }



-- Conformance Information

jmMIBConformance OBJECT IDENTIFIER ::= { jobmonMIB 3 }



-- 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, jmJobIDGroup, jmJobGroup, jmAttributeGroup,
        jmSystemGroup }

    GROUP   jmMirrorAttrGroup
    DESCRIPTION
        "The mirror attribute group (sorted by attribute type).
        Implementation of this group is OPTIONAL.

        An agent that implements the jmMirrorAttrTable SHALL create and
        maintain for the same time a row in the jmMirrorAttrTable for
        each corresponding row in the jmAttributeTable."

    OBJECT   jmGeneralJobSetName
    SYNTAX   JmUTF8StringTC (SIZE(0..8))
    DESCRIPTION
        "Only 8 octets maximum string length NEED be supported by the
        agent."

    OBJECT   jmJobOwner
    SYNTAX   JmJobStringTC (SIZE(0..16))
    DESCRIPTION
        "Only 16 octets maximum string length NEED be supported by the
        agent."

-- There are no CONDITIONALLY MANDATORY groups.

    ::= { jmMIBConformance 1 }
```

```
jmMIBGroups      OBJECT IDENTIFIER ::= { jmMIBConformance 2 }

jmGeneralGroup OBJECT-GROUP
    OBJECTS {
        jmGeneralNumberOfActiveJobs,   jmGeneralOldestActiveJobIndex,
        jmGeneralNewestActiveJobIndex, jmGeneralJobPersistence,
        jmGeneralAttributePersistence, jmGeneralJobSetName}
    STATUS  current
    DESCRIPTION
        "The general group."
    ::= { jmMIBGroups 1 }



jmJobIDGroup OBJECT-GROUP
    OBJECTS {
        jmJobIDJobSetIndex, jmJobIDJobIndex }
    STATUS  current
    DESCRIPTION
        "The job ID group."
    ::= { jmMIBGroups 2 }



jmJobGroup OBJECT-GROUP
    OBJECTS {
        jmJobState, jmJobStateReasons1, jmNumberOfInterveningJobs,
        jmJobKOctetsPerCopyRequested, jmJobKOctetsProcessed,
        jmJobImpressionsPerCopyRequested, jmJobImpressionsCompleted,
        jmJobOwner }
    STATUS  current
    DESCRIPTION
        "The job group."
    ::= { jmMIBGroups 3 }



jmAttributeGroup OBJECT-GROUP
    OBJECTS {
        jmAttributeValueAsInteger, jmAttributeValueAsOctets }
    STATUS  current
    DESCRIPTION
        "The attribute group."
    ::= { jmMIBGroups 4 }
```

```
jmMirrorAttrGroup OBJECT-GROUP
    OBJECTS {
        jmMirrorAttrValueAsInteger, jmMirrorAttrValueAsOctets }
    STATUS  current
    DESCRIPTION
        "The mirror attribute group (sorted by attribute type).
        Implementation of this group is OPTIONAL.

        An agent which implements the jmMirrorAttrTable SHALL create
        and maintain for the same time a row in the jmMirrorAttrTable
        for each corresponding row in the jmAttributeTable."
    ::= { jmMIBGroups 5 }


jmSystemGroup OBJECT-GROUP
    OBJECTS {
        jmSystemVersionString, jmSystemOptionSupport,
        jmSystemAttrIntegerSupport,
        jmSystemAttrOctetsSupport,
        jmSystemAttrMultiRowSupport }
    STATUS  current
    DESCRIPTION
        "The system group."
    ::= { jmMIBGroups 6 }


END
```

5 Appendix A - Implementing the Job Life Cycle

The job object has well-defined states and client operations that affect the transition between the job states. Internal server and device 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 and completed states are required to be implemented by an agent. However, a conforming 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 user visible. The agent SHALL make these states visible in the MIB, but only for the subset of job states that the implementation has. Some implementations MAY need to have sub-states of these user-visible states. The jmJobStateReasons1 object and the jobStateReasons*N* (*N*=2..4) attributes can be used to represent the sub-states of the jobs.

Job states are 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 assume that accounting and auditing logs are separate from the job life cycle and last longer than job entries in the MIB. Jobs in the completed, aborted, or canceled states are not logs, since jobs in these states are accessible via SNMP protocol operations and SHALL be removed from the Job Monitoring MIB tables after a site-settable or implementation-defined period of time. An accounting application MAY copy accounting information incrementally to an accounting log as a job processes, or MAY be copied while the job is in the canceled, aborted, or completed states, depending on implementation. The same is true for auditing logs.

The jmJobState object specifies the standard job states. The normal job state transitions are shown in the state transition diagram presented in Table 1.

6 APPENDIX B - Support of Job Submission Protocols

A companion PWG document, entitled "Job Submission Protocol Mapping Recommendations for the Job Monitoring MIB" [protomap] contains the recommended usage of each of the objects and attributes in this MIB with a number of job submission protocols. In particular, which job submission ID format should be used is indicated for each job submission protocol.

Some job submission protocols have support for the client to specify a job submission ID. A second approach is to enhance the document format to embed the job submission ID in the document data. This second approach is independent of the job submission protocol. This appendix lists some examples of these approaches.

Some PJL implementations wrap a banner page as a PJL job around a job submitted by a client. If this results in multiple job submission IDs, the agent SHALL create multiple jmJobIDEntry rows in the jmJobIDTable that each point to the same job entry in the job tables. See the specification of the jmJobIDEntry.

7 References

[BCP-11] Bradner S., Hovey R., "The Organizations Involved in the IETF Standards Process", 1996/10/29 (RFC 2028)

[GB2312] GB 2312-1980, "Chinese People’s Republic of China (PRC) mixed one byte and two byte coded character set"

[hr-mib] P. Grillo, S. Waldbusser, "Host Resources MIB", RFC 1514, September 1993

[iana] J. Reynolds, and J. Postel, "Assigned Numbers", STD 2, RFC 1700, ISI, October 1994.

[IANA-charsets] Coded Character Sets registered by IANA and assigned an enum value for use in the CodedCharSet textual convention imported from the Printer MIB. See ftp://ftp.isi.edu/in-notes/iana/assignments/character-sets

[iana-media-types] IANA Registration of MIME media types (MIME content types/subtypes). See ftp://ftp.isi.edu/in-notes/iana/assignments/

[ipp-model] Internet Printing Protocol/1.0: Model and Semantics, work in progress on the IETF standards track. See draft-ietf-ipp-model-09.txt. See also http://www.pwg.org/ipp/index.html

[ISO-639] ISO 639:1988 (E/F) - Code for Representation of names of languages - The International Organization for Standardization, 1st edition, 1988.

[ISO-646] ISO/IEC 646:1991, "Information technology -- ISO 7-bit coded character set for information interchange", JTC1/SC2.

[ISO-2022] ISO/IEC 2022:1994 - "Information technology -- Character code structure and extension techniques", JTC1/SC2.

[ISO-3166] ISO 3166:1988 (E/F) - Codes for representation of names of countries - The International Organization for Standardization, 3rd edition, 1988-08-15."

[ISO-8859-1] ISO/IEC 8859-1:1987, "Information technology -- 8-bit single byte coded graphic character sets - Part 1: Latin alphabet No. 1, JTC1/SC2."

[ISO-10646] ISO/IEC 10646-1:1993, "Information technology -- Universal Multiple-Octet Coded Character Set (UCS) - Part 1: Architecture and Basic Multilingual Plane, JTC1/SC2.

[iso-dpa] ISO/IEC 10175-1:1996 "Information technology -- Text and Office Systems -- Document Printing Application (DPA) -- Part 1: Abstract service definition and procedures. See ftp://ftp.pwg.org/pub/pwg/dpa/

[JIS X0208] JIS X0208-1990, "Japanese two byte coded character set."

[mib-II] MIB-II, RFC 1213.

[print-mib] Smith, R., Wright, F., Hastings, T., Zilles, S. and Gyllenskog, J., "Printer MIB", RFC 1759, proposed IETF standard, March 1995. See also [print-mib-draft].

[print-mib-draft] Turner, R., "Printer MIB", work in progress, on the standards track as a draft standard: <draft-ietf-printmib-mib-info-02.txt>, January 22, 1999.

[protomap] Bergman, R., "Job Submission Protocol Mapping Recommendations for the Job Monitoring MIB," work in progress as an informational RFC. See <draft-bergman-printmib-job-protomap-03.txt>, February 10, 1998.

[pwg] The Printer Working Group is a printer industry consortium open to any individuals. For more information, access the PWG web page: http://www.pwg.org

[RFC1738] Berners-Lee, T., Masinter, L., McCahill, M., "Uniform Resource Locators (URL)", RFC 1738, December 1994.

[RFC1766] Avelstrand, H., "Tags for the Identification of Languages", RFC 1766, March 1995.

[RFC2026] S. Bradner, "The Internet Standards Process -- Revision 3", RFC 2026, October 1996.

[RFC2119] S. Bradner, "Keywords for use in RFCs to Indicate Requirement Levels", RFC 2119, March 1997.

[RFC2277] H. Alvestrand, "IETF Policy on Character Sets and Languages" RFC 2277, January 1998.

[RFC2278] N. Freed, J. Postel: "IANA CharSet Registration Procedures", RFC 2278, January 1998.

[SMIv2-SMI] J. Case, et al. "Structure of Management Information for Version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1902, January 1996.

[SMIv2-TC] J. Case, et al. "Textual Conventions for Version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1903, January 1996.

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

[URI-spec] Berners-Lee, T., Fielding, R., Masinter, L., "Uniform Resource Identifiers (URI): Generic Syntax", RFC 2396, August 1998.

[US-ASCII] Coded Character Set - 7-bit American Standard Code for Information Interchange, ANSI X3.4-1986.

[UTF-8] F. Yergeau, "UTF-8, a transformation format of ISO 10646", RFC 2279, January 1998.

8 Notices

9 Author's Addresses

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

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

Tom Hastings
Xerox Corporation, ESAE-231
737 Hawaii St.
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
6300 Diagonal Hwy
Boulder, CO 80301

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

Send questions and comments to the Printer Working Group (PWG) using the Job Monitoring Project (JMP) Mailing List: jmp@pwg.org

To learn how to subscribe, send email to: jmp-request@pwg.org

Implementers of this specification are encouraged to join the jmp mailing list in order to participate in discussions on any clarifications needed and registration proposals for additional attributes and values being reviewed in order to achieve consensus.

For further information, access the PWG web page under "JMP":

    http://www.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 Farrell - Canon
Steve Gebert - IBM
Robert Herriot - Sun Microsystems Inc.
Shige Kanemitsu - Kyocera

David Kellerman - Northlake Software
Rick Landau - Digital
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
Stuart Rowley - Kyocera
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.

10 Change History

This section summarizes the changes in each version after version 1.0 in reverse chronological order.

10.1Changes to produce version 2.0, dated February 20, 1999

The following changes were made to version 1.2, dated October 2, 1998 to make version 2.0, dated February 20, 1999:

1. Added the Mirror table.

2. Moved the JmJobSubmissionIDTypeTC, JmJobStateReasons1TC, JmJobStateReasons2TC, JmJobStateReasons3TC, and JmJobStateReasons4TC assignments out of the MIB and into the Introduction.

3. Added the MANDATORY jmSystemGroup that contains the jmSystemVersionString, jmSystemOptionSupport, jmSystemAttrIntegerSupport, jmSystemAttrOctetsSupport, and jmSystemAttrMultiRowSupport objects.

4. Changed the version number to 2.0, since a MANDATORY table was added.

10.2Changes to produce version 1.2, dated October 2, 1998

The following changes were made to version 1.1, dated October 1, 1998 to make version 1.2, dated October 2, 1998:

1. Removed all REFERENCE clauses since they referred to sections in the specification that were not in the MIB.

2. Moved the definitions of the attributes from the TC to a new section 3.3.8 as requested by the IESG.

3. Removed the attributes from the Table of Contents

4. Added the data types as ASN.1 comments after each attribute enum.

5. Changed a number of occurrences of "SHALL" to "is" when they were just definitions, rather than conformance requirements.

10.3Changes to produce version 1.1, dated October 1, 1998

The following changes were made to version 1.0, dated February 3, 1998 to make version 1.1, dated October 1, 1998:

1. Clarified sections 3.3.3 and 3.3.7 so that the DEFVAL of 0 for index attributes is different from the DEFVAL for jmAttributeValueAsInteger which is -2.

2. Clarified the relationships of the values of the JmJobCollationTypeTC with the IPP "multiple-document-handling" attribute.

3. Clarified that the values of the mediumRequested(170) and mediumConsumed(171) attributes may be any of the IPP 'media' values which are media names, media size names, and input tray names.

4. Added the two attributes approved by the PWG for registration in April 1998: mediumTypeConsumed(174) and mediumSizeConsumed(175).

5. Changed "insure" to "ensure'.

6. Correct an incorrect reference in the jmAttributeEntry DESCRIPTION from jmJobTable to jmAttributeTable.

11 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 no special prefix.