Job Monitoring MIB

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

From: Tom Hastings
Date: 06/09/97
Version: 0.82
File: ftp://ftp.pwg.org/pub/jmp/mibs/jmp-mib.doc .pdf jmp-mibr.doc .pdf .pdr

Status: Fifth draft MIB that corresponds to the changes agreed to at the JMP meeting, on Friday, 5/16/97. The major changes were to eliminated duplicates between the Job State table and the Attribute table, and to move all mandatory integer attributes to the Job Table, leaving only the **jobOwner** string attribute as MANDATORY in the Attribute table. The Job State table has been renamed back to the Job table, since it has more than just state now. See the change history in the separate file: changes.doc .pdf.

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

The MIB has been greatly simplified so that now there are only 17 objects in the MIB. There are 71 attributes: 1 is MANDATORY and 70 are OPTIONAL.

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

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

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

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

INTERNET-DRAFT Ron Bergman
Dataproducts Corp.
Tom Hastings
Xerox Corporation
Scott Isaacson
Novell, Inc.
Harry Lewis
IBM Corp.
April 1997

Job Monitoring MIB - V0.82

<draft-ietf-printmib-job-monitor-01.txt>

Expires Dec 10, 1997

Status of this Memo

Abstract

This Internet-Draft specifies a set of 17 SNMP MIB objects for (1) monitoring the status and progress of print jobs (2) obtaining resource requirements before a job is processed, (3) monitoring resource consumption while a job is being processed and (4) collecting resource accounting data after the completion of a job. This MIB is intended to be implemented (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

The Job Monitoring MIB consists of a 6-object General Group, a 2-object Job Submission ID Group, a 7-object Job Group, and a 2-object Attribute Group. Each group is a table. 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. One MANDATORY attribute and 70 OPTIONAL attributes are defined as textual conventions.

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

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[1] (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 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 finishing 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 co-located with the printer 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 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 SHALL 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[2] and the Internet Printing Protocol (IPP)[3], 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)[4], as well as with ones that do implement ISO DPA. So nothing in the job monitoring MIB requires implementation of 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.

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

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

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

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

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

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

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

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

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

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

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

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

A *monitor* or *job monitoring application* is the network entity that End Users, System Operators, Accountants, Asset Managers, and Capacity Planners use to monitor jobs using

SNMP. A monitor MAY be either a separate application or MAY be part of the client that also submits jobs.

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

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

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

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

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

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

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

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

An *SNMP information object* is 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.

An *attribute* is a name, value-pair that specifies an 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 SHALL 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.

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

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

3. 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 Goals section.

The diagram in the Printer MIB[1] 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.

3.1 Configuration 1 - client-printer

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

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

The job submitting **client** and/or **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, and longer in order to implement the **completed** state in which monitoring programs can copy out the accounting data from the Job Monitoring MIB.

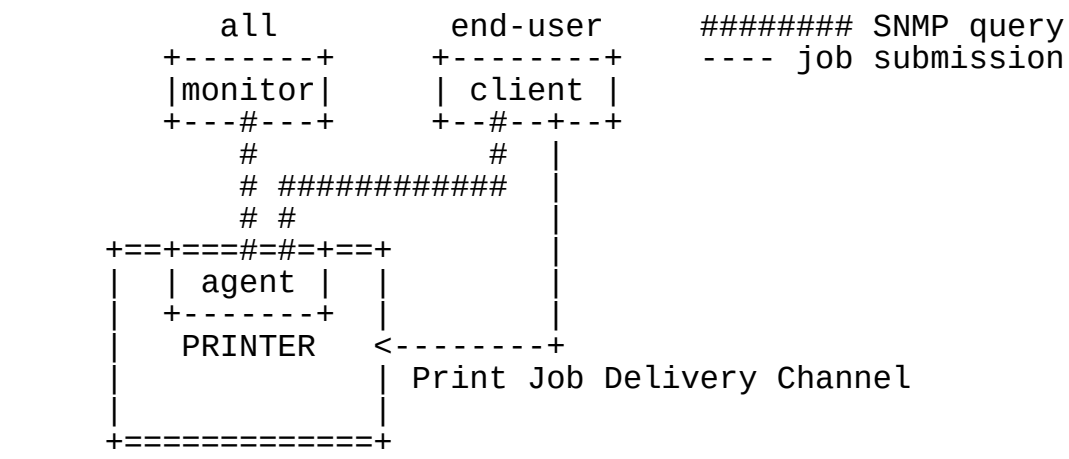


Figure 3-1 - Configuration 1 - client-printer - agent in the printer

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

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

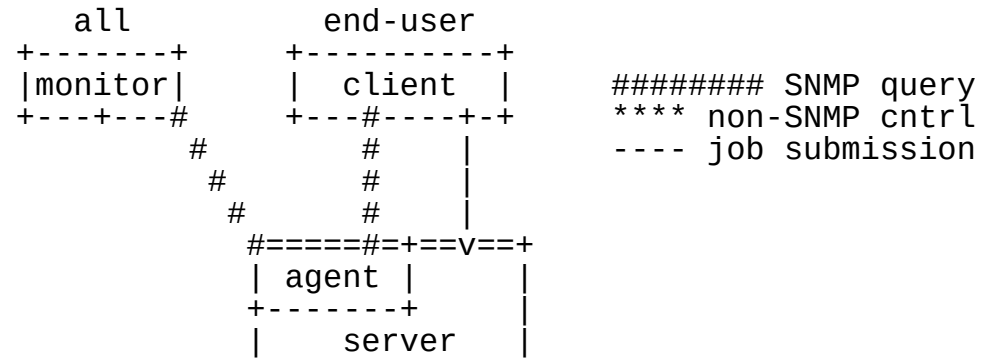
3.2 Configuration 2 - client-server-printer - agent in the server

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

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

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



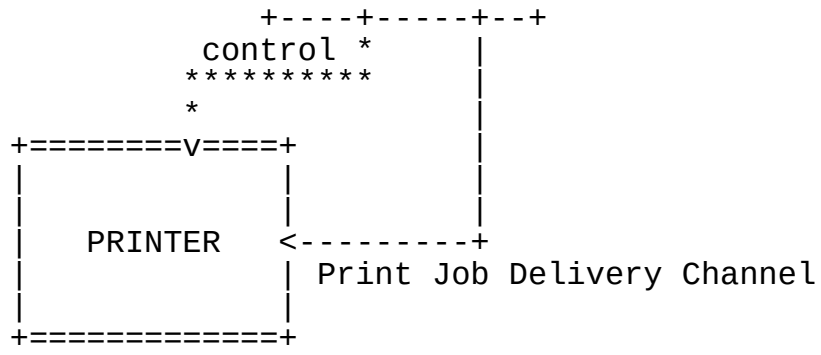


Figure 3-2 - Configuration 2 - client-server-printer - agent in the server

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

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

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

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

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

1. the server using some protocol to monitor jobs in the server that does not contain the Job Monitoring MIB AND
2. a Job Monitoring MIB agent that is part of the **printer** to monitor jobs after the server passes the jobs to the printer. In such configurations, the server deletes its copy of the job from the server after submitting the job to the printer usually almost immediately (before the job does much processing, if any)**.**

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

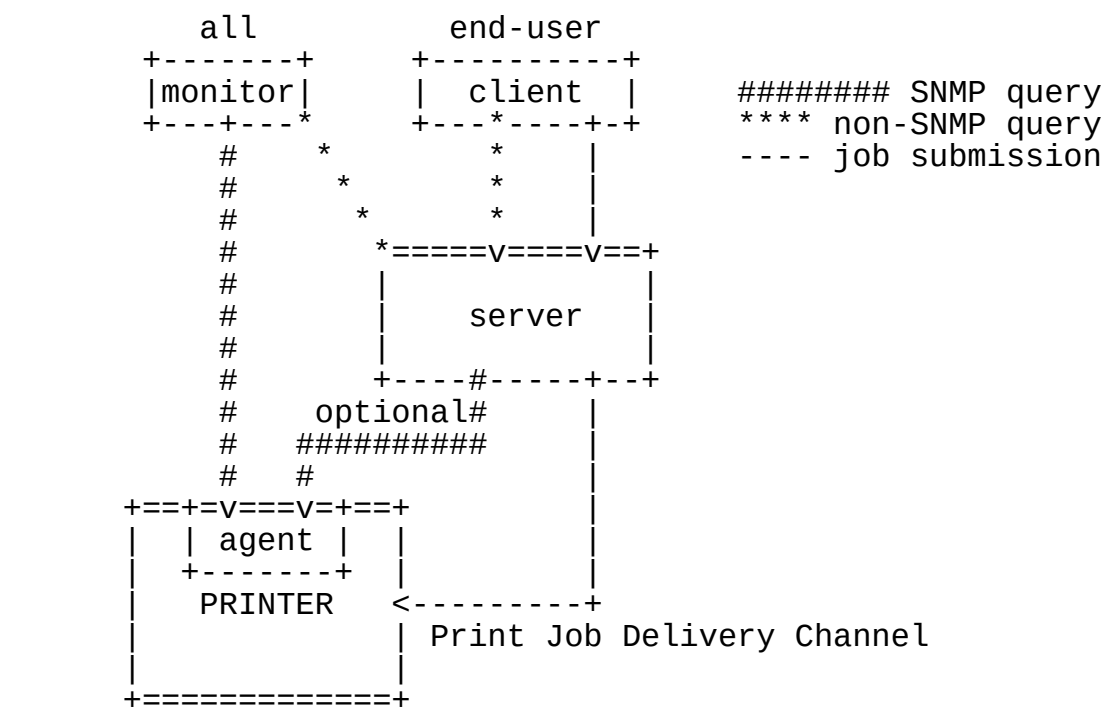


Figure 3-3 - Configuration 3 - client-server-printer - client monitors printer agent and server

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

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**.

4. Conformance Considerations

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

4.1 Conformance Terminology

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

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

4.2 Agent Conformance Requirements

A conforming agent:

1. SHALL implement *all* MANDATORY groups and attributes in this specification.
2. NEED NOT implement any OPTIONAL attributes, whether the agent is able to obtain the information from the server or device.
3. NEED NOT implement both forms of an attribute if it implements an attribute that permits a choice of Integer and Octets forms, though implementing both forms may help management applications by giving them a choice of representations, since the representation are equivalent. See page 46.

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

4.2.1 MIB II System Group objects

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

4.2.2 MIB II Interface Group objects

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

4.2.3 Printer MIB objects

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

4.3 Job Monitoring Application Conformance Requirements

A conforming job monitoring application:

1. SHALL accept all objects in all MANDATORY groups and all MANDATORY attributes that are required to be implemented by an agent according to Section 4.2 and SHALL either present them to the user or ignore them.
2. SHALL accept *all* OPTIONAL attributes, including enum and bit values specified in this specification and additional ones that may be registered with IANA 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 7 entitled "IANA Considerations" on page 21.
3. SHALL accept either form of time attribute, if it supports a time attribute, since agents are free to implement either time form. See page 46.

5. Job Identification

There are a number of attributes that permit a user, operator or system administrator to identify jobs of interest, such as **jobOwner**, **jobName**, etc. In addition, there is a Job Submission ID object that 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. 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 Job Submission ID allows the monitoring application to identify the job of interest from all the jobs currently "known" by the server or device. The Job Submission ID can be assigned by either the client's local system or a downstream server or device. The point of assignment will be determined by the job submission protocol in use.

The server/device-side identifier, called the **jmJobIndex** object, will be assigned by the server or device that accepts the jobs from submitting clients. The MIB agent SHALL use the job identifier assigned by the server or device to the job as the value of the **jmJobIndex** object that defines the table rows (there are multiple tables) that contain the information relating to the job. This object allows the interested party to obtain all objects desired that relate to this job. The MIB provides a mapping table that maps each Job Submission ID to the corresponding **jmJobIndex** value, so that an application can determine the correct value for the jmJobIndex value for the job of interest in a single Get operation. See the **jmJobIDGroup** on page 60.

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.

6. Internationalization Considerations

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

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

is running in, let alone contain an object to control what locale the agent is to use to represent coded character set objects.

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

7. IANA Considerations

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

7.1 IANA Registration of enums

This specification uses textual conventions to define enumerated values (enums) 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.

This working group has defined several type of enumerations for use in the Job Monitoring MIB and the Printer MIB[1]. 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:

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.

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

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 after review by this working group. The initial versions of the MIB will contain the values registered so far. After the

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

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

1. **JmTimeStampTC**
2. **JmFinishingTC**
3. **JmPrintQualityTC**
4. **JmTonerEconomyTC**
5. **JmPrinterResolutionTC**
6. **JmMediumTypeTC**
7. **JmJobStateTC**
8. **JmAttributeTypeTC**

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 without working group review. The initial versions of the MIB will contain the values registered so far. After the MIB is approved, additional values will be registered through IANA without approval by this working group.

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

7.2 IANA Registration of 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 SHALL follow the procedures for a type 2 enum as specified in Section 7.1.2.

7.3 IANA Registration of Job Submission Id Formats

In addition to enums and bit values, this specification assigns numbers to various job submission ID formats. See **jmJobSubmissionID** on page 61. The registration of

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

720 8. Security Considerations

721 8.1 Read-Write objects

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

726 8.2 Read-Only Objects In Other User's Jobs

727 The security policy of some sites may be that unprivileged users can only get the objects
728 from jobs that they submitted, plus a few minimal objects from other jobs, such as the
729 **jmJobKOctetsRequested** and **jmJobKOctetsCompleted** objects**,** so that a user can tell
730 how busy a printer is. Other sites might allow all unprivileged users to see all objects of
731 all jobs. It is up to the agent to implement any such restrictions based on the identification
732 of the user making the SNMP request. This MIB does not require, nor does it specify
733 how, such restrictions would be implemented. A monitoring application SHOULD
734 enforce the site security policy with respect to returning information to an unprivileged
735 end user that is using the monitoring application to monitor jobs that do not belong to that
736 user, i.e., the **jobOwner** attribute in the **jmAttributeTable** does not match the user's user
737 name. See the **JmAttributeTypeTC** textual convention on page 33 and the
738 **jmAttributeTable** on page 67.

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

741 9. Returning Objects With No Value In Mandatory Groups

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

747 10. Notification and Traps

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

11. MIB specification

The following pages constitute the actual Job Monitoring MIB.

```
752  Job-Monitoring-MIB DEFINITIONS ::= BEGIN
753
754  IMPORTS
         MODULE-IDENTITY, OBJECT-TYPE, experimental, Integer32
                                                          FROM SNMPv2-SMI
         TEXTUAL-CONVENTION                               FROM SNMPv2-TC
         MODULE-COMPLIANCE, OBJECT-GROUP                  FROM SNMPv2-CONF;
         -- The following textual-conventions are needed
         -- to implement certain attributes, but are *not*
         -- needed to compile this MIB.  They are
         -- provided here for convenience:
         -- DateAndTime                                   FROM SNMPv2-TC
         -- PrtAlertCodeTC, PrtInterpreterLangFamilyTC    FROM Printer-MIB
755
756  -- Use the experimental (54) OID assigned to the Printer MIB[1] before
757  -- it was published as RFC 1759.
758  -- Upon publication of the Job Monitoring MIB as an RFC, delete this
759  -- comment and the line following this comment and change the
760  -- reference of { temp 104 } (below) to { mib-2 X }.
761  -- This will result in changing:
762  -- 1 3 6 1 3 54 jobmonMIB(105)    to:
763  -- 1 3 6 1 2  1 jobmonMIB(X)
764  -- This will make it easier to translate prototypes to
765  -- the standard namespace because the lengths of the OIDs won't
766  -- change.
767  temp OBJECT IDENTIFIER ::= { experimental 54 }
768
769  jobmonMIB MODULE-IDENTITY
770          LAST-UPDATED "9705200000Z"
771          ORGANIZATION "IETF Printer MIB Working Group"
772          CONTACT-INFO
773                  "Tom Hastings
774                  Postal:  Xerox Corp.
775                          Mail stop ESAE-231
776                          701 S. Aviation Blvd.
777                          El Segundo, CA 90245
778
779                  Tel:     (301)333-6413
780                  Fax:     (301)333-5514
781                  E-mail:  hastings@cp10.es.xerox.com"
782          DESCRIPTION
783                  "The MIB module for monitoring job in servers, printers, and other devices.
784
785                  File: jmp-mib.doc,  .pdf,  .txt,  .mib
786                  Version: 0.82"
787          ::= { temp 105 }
788
789
790
```

```
-- Textual conventions for this MIB module


JmTimeStampTC ::= TEXTUAL-CONVENTION
        STATUS      current
        DESCRIPTION
                "The simple time at which an event took place.  The units SHALL be 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[5].)

                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 (see page 68).  The
                TimeStamp textual-convention defined in SMNPv2-TC is defined as an APPLICATION 3
                IMPLICIT INTEGER tag, not an Integer32, 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 7.1 on page 21.
        SYNTAX      INTEGER {
                      other(1),
                      unknown(2),
                      sptUNIX(3),              -- UNIX(tm)
                      sptOS2(4),               -- OS/2
                      sptPCDOS(5),             -- DOS
                      sptNT(6),                -- NT
                      sptMVS(7),               -- MVS
                      sptVM(8),                -- VM
                      sptOS400(9),             -- OS/400
                      sptVMS(10),              -- VMS
                      sptWindows95(11),        -- Windows95
                      sptNetWare(33)           -- NetWare
        }
```

826
827
828 **JmFinishingTC** ::= TEXTUAL-CONVENTION
829 STATUS current
830 DESCRIPTION
831 "The type of finishing."
832
833 -- This is a type 2 enumeration. See Section 7.1 on page 21.
834 SYNTAX INTEGER {

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

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

none(3),
-- Perform no finishing.

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

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

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

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

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

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

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

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

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

                **bind(13)**
                    --  This value indicates that a binding is to be applied to the document; the type and
                    --  placement of the binding is site-defined.

835         }
836
837
838
839
840
841     **JmPrintQualityTC** ::= TEXTUAL-CONVENTION
842         STATUS      current
843         DESCRIPTION
844             "Print quality settings."
845
846         -- This is a type 2 enumeration.  See Section 7.1 on page 21.
847         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.
848         }
849
850
851     **JmPrinterResolutionTC** ::= TEXTUAL-CONVENTION
852         STATUS      current
853         DESCRIPTION
854             "Printer resolutions.
855
856             The values are type2 enums that represent single integers or pairs of integers.  The latter are to
857             specify the resolution when the x and y dimensions differ.  When two integers are specified, the
858             first is in the x direction, i.e., in the direction of the shortest dimension of the medium, so that
859             the value is independent of whether the printer feeds long edge or short edge first."
860
861         -- This is a type 2 enumeration.  See Section 7.1 on page 21.
862         SYNTAX      INTEGER {
                **other(1),**          --  Not one of the specified or registered values.
                **unknown(2),**        --  The actual value is unknown.
                **normal(3),**         --  Normal resolution.
                **res100(4),**         --  100 x 100 dpi
                **res200(5),**         --  200 x 200 dpi
```

```
                **res240(6),**          --  240 x 240 dpi
                **res300(7),**          --  300 x 300 dpi
                **res360(8),**          --  360 x 360 dpi
                **res600(9),**          --  600 x 600 dpi
                **res720(10),**         --  720 x 720 dpi
                **res800(11),**         --  800 x 800 dpi
                **res1200(12),**        --  1200 x 1200 dpi
                **res1440(13),**        --  1440 x 1440 dpi
                **res1800(14),**        --  1800 x 1800 dpi
                **res100x200(15),**     --  100 x 200 dpi
                **res300x600(16),**     --  300 x 600 dpi
                **res600x300(17),**     --  600 x 300 dpi
                **res360x720(18),**     --  360 x 720 dpi
                **res720x360(19),**     --  720 x 360 dpi
                **res400x800(20),**     --  400 x 800 dpi
                **res800x400(21),**     --  800 x 400 dpi
                **res600x1200(22),**    --  600 x 1200 dpi
                **res1200x600(23),**    --  1200 x 600 dpi
                **res720x1440(24),**    --  720 x 1440 dpi
                **res1440x720(25),**    --  1440 x 720 dpi
                **res1800x600(26)**     --  1800 x 600 dpi
        }


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

        -- This is a type 2 enumeration.  See Section 7.1 on page 21.
        SYNTAX      INTEGER {
                **off(0),**             --  Off.  Normal.  Use full toner.
                **on(1)**               --  On.  Use less toner than normal.
        }


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

```
887         -- This is a type 2 enumeration.  See Section 7.1 on page 21.
888         SYNTAX      INTEGER {
              other(1),
                    --  The type is neither one of the values listed in this specification nor a registered value.
                    --
              unknown(2),
                    --  The type is not known.

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

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

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

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

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

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

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

              tabStock(10),
                    --  Media with tabs.

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

              labels(12),
                    --  Label-stock.

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

889         }
890
891
892
893
```

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:

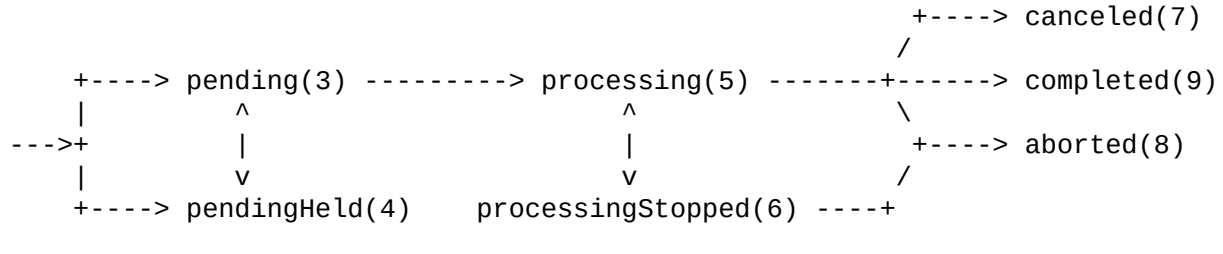


Figure 4 - Normal Job State Transitions

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**, **processing**, and **processingStopped** states.

Jobs in the **pending**, **processing**, and **processingStopped** states are called 'active', while jobs in the **pendingHeld**, **canceled**, **aborted**, and **completed** are called 'in-active'."

-- This is a type 2 enumeration. See Section 7.1 on page 21.
SYNTAX INTEGER {
other(1),
-- The job state is *not* one of the defined states.

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

pending(3),
-- 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 **jobStateReasons*n*** (***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 **jobStateReasons*n*** (*n*=2..4) attributes. See the
-- **JmJobStateReasons*n*TC** (*n*=1..4) textual convention on page (49) for the
-- specification of each reason.

processing(5),

-- Either:
--
-- 1. The job is using, or is attempting to use, one or more document
-- transforms which include (1) purely software processes that are interpreting
-- a PDL, and (2) hardware devices that are interpreting a PDL, making marks
-- on a medium, and/or performing finishing, such as stapling, etc.
--
-- OR
--
-- 2. (configuration 2) the server has made the job ready for printing, but the
-- output device is not yet printing it, either because the job hasn't reached the
-- output device or because the job is queued in the output device or some
-- other spooler, awaiting the output device to print it.
--
-- 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.

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 indicate its
-- condition in human readable form locally at the device. The management
-- application can obtain more complete device status remotely by querying the
-- appropriate device MIB using the job's **deviceIndex** attribute(s), if the agent
-- implements such a device MIB

canceled(7),
-- A client has canceled the job and the job is either: (1) in the process of being
-- terminated by the server or device or (2) has completed terminating. The
-- job's **jmJobStateReasons1** object SHOULD contain either the
-- **canceledByUser** or **canceledByOperator** value.

aborted(8),
-- The job has been aborted by the system, usually while the job was in the
-- **processing** or **processingStopped** 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). The job's **jmJobStateReasons1** object SHOULD
-- contain one of: **completedSuccessfully**, **completedWithWarnings**, or
-- **completedWithErrors** values.

}

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

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

Most attributes apply to all three configurations covered by this MIB specification (see section 3 on page 14). Those attribute that apply to a particular configuration are indicated as '**Configuration *n*:**'.

Conformance of Attribute Implementation

A very few attributes are MANDATORY for conformance, and the rest are OPTIONAL. An agent SHALL instrument any MANDATORY attribute. If the server or device does not provide access to the information about the MANDATORY attribute, the agent SHALL return the '**unknown**' value. For attributes represented by a counting integer, the unknown value is (**-2**) and for attributes represented by an enum, the unknown value is (**2**), as in the Printer MIB[1]. For attributes represented by an OCTET STRING, the unknown value is a zero-length string, unless specified otherwise.

The MANDATORY attributes are:

jobOwner(20)

The attributes not labeled as MANDATORY are OPTIONAL. An agent MAY, but NEED NOT, implement any OPTIONAL attributes.

NOTE - The table of contents lists all the attributes in order to help see the order of enum assignments which is the order that the GetNext operation can be used to get attributes. The table of contents also indicates the MANDATORY attributes as: (MANDATORY).

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.

Datatypes and Attribute Naming Conventions

The datatype of each attribute is indicated on the first line(s) of the description. Some attributes have several different data type representations. When the data types can be represented in a single row in the **jmAttributeTable**, the data type name is not included as the last part of the name of the attribute. When the data 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 datatype is the last part of the name of the attribute: **Name**, **Index**, **DateAndTime**, **TimeStamp**, etc.

NOTE: No attribute name exceeds 31 characters.

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

Most attributes SHALL 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 **JmAttributeTypeTC**, an agent SHALL ensure that each attribute item occurs only once in the **jmAttributeTable** for a job. Attributes that are permitted to appear multiple times in the **jmAttributeTable** for a job are indicated with '**MULTI-ROW:**' in their specification in the **JmAttributeTypeTC**. However, such '**MULTI-ROW**' attribute items SHALL *not* contain duplicates for 'intensive' (as opposed to 'extensive') attributes.

> For example, a job or document(s) may use multiple PDLs. However, each distinct **documentFormat** attribute value SHALL appear in the **jmAttributeTable** only once for a job since the interpreter language is an intensive attribute item, even though the job has a number of documents that all use the same PDL.
>
> As another example of an intensive attribute that can have multiple entries, if a document or job uses multiple types of media, there SHALL be only one row in the **jmAttributeTable** for each media type, not one row for each document that uses that medium type.
>
> On the other hand, if a job contains two documents of the same name, there can be separate rows for the **documentName** attribute item with the same name, since a document name is an extensive attribute item. The specification indicates that the values NEED NOT be unique for such '**MULTI-ROW:** attributes'

Value Represented As Integer Or Octets

In the following definitions of the enums, each description indicates whether the value of the attribute SHALL be represented using the **jmAttributeValueAsInteger** or the **jmAttributeValueAsOctets** objects by the initial tag: '**INTEGER:**' or '**OCTETS:**', respectively. Some attributes allow the agent a choice of either an integer and/or an octets representation, 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 values (see **mediumConsumed(171)**). These attributes are indicated with '**INTEGER:**' and/or '**OCTETS:**' tags. See the **jmAttributeGroup** starting on page 66 for the descriptions of these objects.

Consumption Attributes

A number of 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 **xxxxYyyyCompleted** and **xxxxYyyyConsumed** attribute specification.

Index Value Attributes

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

The standard attribute types defined so far are:"

```
-- This is a type 2 enumeration.  See Section 7.1 on page 21.
SYNTAX      INTEGER {
  -- jmAttributeTypeIndex                          Datatype
       --  Description - including 'OCTETS:' or 'INTEGER:' to specify whether the value
       --  SHALL be represented in the jmAttributeValueAsOctets or the
       --  jmAttributeValueAsInteger object, or both, respectively.

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

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


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

jobStateReasons2(3), -- **JmJobStateReasons2TC** (pg 52)
-- INTEGER: Additional information about the job's current state that augments the
-- **jmJobState** object. See the description under the **JmJobStateReasons1TC** textual-
-- convention on page 49.

jobStateReasons3(4), -- **JmJobStateReasons3TC** (pg 55)
-- INTEGER: Additional information about the job's current state that augments the
-- **jmJobState** object. See the description under **JmJobStateReasons1TC** textual-
-- convention on page 49.

jobStateReasons4(5), -- **JmJobStateReasons4TC** (pg 55)
-- INTEGER: Additional information about the job's current state that augments the
-- **jmJobState** object. See the description under **JmJobStateReasons1TC** textual-
-- convention on page 49.

deviceAlertCode(6), -- **PrtAlertCodeTC** (Printer-MIB)
-- INTEGER: The device alert code when the job is stopped because the device needs
-- attention, i.e., needs human intervention. When the device is a printer, this device
-- alert code SHALL be the printer alert code defined by the Printer MIB[1] using the
-- **PrtAlertCodeTC** textual convention or equivalent.

processingMessage(7), -- **OCTET STRING(SIZE(0..63))**
-- OCTETS: MULTI-ROW: A coded character set message that is generated during
-- the processing of the job as a simple form of processing log to show progress and any
-- problems.
--
-- There is no restriction on the same message in multiple rows.

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

jobOwner(20), -- **OCTET STRING(SIZE(0..63))**
-- (MANDATORY)
-- OCTETS: The coded character set name of the user that submitted the job. The
-- method of assigning this user name will be system and/or site specific but the method
-- must insure that the name is unique to the network that is visible to the client and
-- target device.
--
-- This value SHOULD be the *authenticated* name of the user submitting the job.
--
-- In order to assist users to find their jobs for job submission protocols that don't supply
-- a **jmJobSubmissionID**, the agent SHOULD maintain the **jobOwner** attribute for the
-- time specified by the **jmGeneralJobPersistence** object, rather than the (shorter)

-- **jmGeneralAttributePersistence** object.

jobAccountName(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.
--
-- NOTE: This attribute NEED NOT be printable characters.

serverAssignedJobName(22), -- **OCTET STRING(SIZE(0..63))**
-- OCTETS: Configuration 3 only: The human readable string name of the job as
-- assigned by the server that submitted the job to the device that the agent in
-- instrumenting with this MIB.
--
-- NOTE - This attribute is intended for enabling a user to find his/her job that a server
-- submitted to a device after the user submitted the job to the server when the
-- **jmJobSubmissionID** is not supported by the job submission protocol.

jobName(23), -- **OCTET STRING(SIZE(0..63))**
-- OCTETS: The human readable string name of the job as assigned by the submitting
-- user to help the user distinguish between his/her various jobs. This name does not
-- need to be unique.
--
-- This attribute is intended for enabling a user or the user's application to convey a job
-- name that MAY be printed on a start sheet, returned in a **query** result, or used in
-- notification or logging messages.
--
-- In order to assist users to find their jobs for job submission protocols that don't supply
-- a **jmJobSubmissionID**, the agent SHOULD maintain the **jobName** attribute for the
-- time specified by the **jmGeneralJobPersistence** object, rather than the (shorter)
-- **jmGeneralAttributePersistence** object.
--
-- If this attribute is not specified when the job is submitted, no job name is assumed, but
-- implementation specific defaults are allowed, such as the value of the
-- **documentName** attribute of 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** (pg 48)
-- 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** + 0**x**4, respectively, yielding: **0x2C**.
--
-- Whether this attribute is set from a job attribute supplied by the job submission client
-- or is set by the recipient job submission server or device depends on the job
-- submission protocol. This attribute SHALL be implemented if the server or device
-- has other types in addition to or instead of printing.
--
-- One of the purposes of this attribute is to permit a requester to filter out jobs that are
-- not of interest. For example, a printer operator may only be interested in jobs that
-- include printing. That is why this attribute is in the job identification category.

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

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

submittingServerName(27), -- **OCTET STRING(SIZE(0..63))**
-- OCTETS: For configuration 3 only: The administrative name of the server that
-- submitted the job to the device.

submittingApplicationName(28), -- **OCTET STRING(SIZE(0..63))**
-- OCTETS: The name of the client application (not the server in configuration 3) that
-- submitted the job to the server or device.

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

deviceNameRequested(30), -- **OCTET STRING(SIZE(0..63))**
-- OCTETS: The administratively defined coded character set name of the target device
-- requested by the submitting user. For configuration 1, its value corresponds to the
-- Printer MIB[1]: **prtGeneralPrinterName** object (added to the draft Printer MIB) for
-- printers. For configuration 2 and 3, its value is the name of the logical or physical

-- device that the user supplied to indicate to the server on which device(s) they wanted
-- the job to be processed.

queueNameRequested(31), -- **OCTET STRING(SIZE(0..63))**
-- OCTETS: The administratively defined coded character set name of the target queue
-- requested by the submitting user. For configuration 1, its value corresponds to the
-- queue in the device that the agent is instrumenting. For configuration 2 and 3, its
-- value is the name of the queue that the user supplied to indicate to the server on
-- which device(s) they wanted the job to be processed.
--
-- NOTE - typically an implementation SHOULD support either the
-- **deviceNameRequested** or **queueNameRequested** attribute, but not both.

physicalDevice(32), -- **hrDeviceIndex** (see HR MIB)
-- AND/OR
-- **OCTET STRING(SIZE(0..63))**
-- INTEGER: MULTI-ROW: The index of the physical device MIB instance
-- requested/used, such as the Printer MIB[1]. This value is an **hrDeviceIndex** value.
-- See the Host Resources MIB[6].
-- AND/OR
-- OCTETS: MULTI-ROW: The name of the physical device to which the job is
-- assigned.

numberOfDocuments(33), -- **Integer32(0..2147483647)**
-- INTEGER: The number of documents in this job. If this attribute is not present, the
-- number of documents SHALL be 1.

fileName(34), -- **OCTET STRING(SIZE(0..63))**
-- OCTETS: MULTI-ROW: The coded character set file name of the document.
--
-- There is no restriction on the same file name in multiple rows.

documentName(35), -- **OCTET STRING(SIZE(0..63))**
-- OCTETS: MULTI-ROW: The coded character set name of the document.
--
-- There is no restriction on the same document name in multiple rows.

jobComment(36), -- **OCTET STRING(SIZE(0..63))**
-- OCTETS: An arbitrary human-readable coded character text string supplied by the
-- submitting user or the job submitting application program for any purpose. For
-- example, a user might indicate what he/she is going to do with the printed output or
-- the job submitting application program might indicate how the document was
-- produced.
--
-- The **jobComment** attribute is not intended to be a name; see the **jobName** attribute.

documentFormatIndex(37), -- **Integer32(0..2147483647)**
-- INTEGER: MULTI-ROW: The index in the **prtInterpreterTable** in the Printer
-- MIB[1] 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 attribute items 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[1] prtInterpreterLangFamily object, that this job requires/uses.  A
        --  document or a job MAY use more than one PDL or control language.
        --
        --  NOTE - This attribute is represented by a type 2 enum defined in the draft Printer
        --  MIB[1], but is not in RFC 1759.
        --
        --  AND/OR
        --
        --  OCTETS:  MULTI-ROW:  The document format registered as a MIME type, i.e., the
        --  name of the MIME type.
        --
        --  NOTE - IPP[3] uses MIME type keywords to identify document formats.


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

jobPriority(50),                        --  Integer32(1..100)
        --  INTEGER:  The priority for scheduling the job. It is used by servers and devices that
        --  employ a priority-based scheduling algorithm.
        --
        --  A higher value specifies a higher priority. The value 1 is defined to indicate the lowest
        --  possible priority (a job which a priority-based scheduling algorithm SHALL pass over
        --  in favor of higher priority jobs). The value 100 is defined to indicate the highest
        --  possible priority. Priority is expected to be evenly or 'normally' distributed across this
        --  range. The mapping of vendor-defined priority over this range is implementation-
        --  specific.

jobProcessAfterDateAndTime(51),         --  DateAndTime (SNMPv2-TC)
```

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

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

jobHoldUntil(53), -- **OCTET STRING(SIZE(0..63))**
-- OCTETS: The named time period during which the job SHALL become a candidate
-- for processing, such as 'no-hold', 'evening', 'night', 'weekend', 'second-shift', 'third-
-- shift', etc., as defined by the system administrator. Until that time period arrives, the
-- job SHALL be in the **pendingHeld** state with the **jobHoldUntilSpecified** value in
-- the **jmJobStateReasons1** object.

outputBin(54), -- **Integer32(0..2147483647)**
-- AND/OR
-- **OCTET STRING(SIZE(0..63))**
-- INTEGER: MULTI-ROW: The output subunit index in the Printer MIB[1]
-- AND/OR
-- OCTETS: the name of the output bin to which all or part of the job is placed in.

sides(55), -- **Integer32(-2..1)**
-- INTEGER: MULTI-ROW: The number of sides that any document in this job
-- requires/used.

finishing(56), -- **JmFinishingTC** (pg 27)
-- INTEGER: MULTI-ROW: Type of finishing that any document in this job
-- requires/used.

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

printQualityRequested(70), -- **JmPrintQualityTC** (pg 28)
-- INTEGER: MULTI-ROW: The print quality selection requested for document in the
-- job for printers that allow quality differentiation.

printQualityUsed(71), -- **JmPrintQualityTC** (pg 28)
-- INTEGER: MULTI-ROW: The print quality selection actually used by documents in
-- the job for printers that allow quality differentiation.

printerResolutionRequested(72), -- **JmPrinterResolutionTC**
-- (pg 28)
-- INTEGER: MULTI-ROW: The print quality selection requested for document in the
-- job for printers that allow quality differentiation.

printerResolutionUsed(73), -- **JmPrinterResolutionTC**
-- (pg 28)
-- INTEGER: MULTI-ROW: The print quality selection actually used by documents in
-- the job for printers that allow quality differentiation.

tonerEcomonyRequested(74), -- **JmTonerEconomyTC** (pg 29)
-- INTEGER: MULTI-ROW: The print quality selection requested for documents in
-- the job for printers that allow toner quality differentiation.

tonerEcomonyUsed(75), -- **JmTonerEconomyTC** (pg 29)
-- INTEGER: MULTI-ROW: The print quality selection actually used by documents in
-- the job for printers that allow toner quality differentiation.

tonerDensityRequested(76), -- **Integer32(1..20)**
-- INTEGER: MULTI-ROW: The toner density requested for documents in this job
-- for devices that can vary toner density levels. Level 1 is the lowest density and level
-- 20 is the highest density level. Devices with a smaller range, SHALL map the 1-20
-- range evenly onto the implemented range.

tonerDensityUsed(77), -- **Integer32(1..20)**
-- INTEGER: MULTI-ROW: The toner density used by documents in this job for
-- devices that can vary toner density levels. Level 1 is the lowest density and level 20 is
-- the highest density level. Devices with a smaller range, SHALL map the 1-20 range
-- evenly onto the implemented range.

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

jobCopiesRequested(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. 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 that
-- the agent is instrumenting. This count is independent of the number of copies of the
-- job or documents that will be produced, but is just a measure of the number of bytes
-- transferred to the server or device.
--
-- The agent SHALL round the actual number of octets transferred 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
-- **jmJobKOctetsRequested** object and the **jobKOctetsTransferred** attribute SHALL
-- be equal.
--
-- NOTE - The **jobKOctetsTransferred** can be used in the numerator with the
-- **jmJobKOctetsRequested** object in the denominator in order to produce a
-- "thermometer" that indicates the progress of the job for agents that do not implement
-- the **jmJobKOctetsProcessed** object.

-- ++
-- Impression attributes
--
-- For a print job, an impression is the marking of the
-- entire side of a sheet. Two-sided processing involves two
-- impressions per sheet. Two-up is the placement of two
-- logical pages on one side of a sheet and so is still a
-- single impression. See also **jmJobImpressionsRequested** and
-- **jmJobImpressionsCompleted** objects in the **jmJobTable** on page
-- **66**.
-- ++

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.

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

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

pagesRequested(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.

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

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

sheetsCompleted(151),                   -- Integer32(-2..2147483647)
        -- INTEGER:  The 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 reset to 0 as each document in the job starts
        -- being processed and for each document copy as it starts being processed.


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

mediumRequested(170),                   -- JmMediumTypeTC (pg 29)
                                        -- AND/OR
                                        -- OCTET STRING(SIZE(0..63))
        -- INTEGER:  MULTI-ROW:  The type
        -- AND/OR
        -- OCTETS:  the name of the medium that is required by the job.
```

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

colorantRequested(172), -- **Integer32(0..2147483647)**
-- AND/OR
-- **OCTET STRING(SIZE(0..63))**
-- INTEGER: MULTI-ROW: The index (**prtMarkerColorantIndex**) in the Printer
-- MIB[1]
-- AND/OR
-- OCTETS: the name of the colorant requested.

colorantConsumed(173), -- **Integer32(0..2147483647)**
-- AND/OR
-- **OCTET STRING(SIZE(0..63))**
-- INTEGER: MULTI-ROW: The index (**prtMarkerColorantIndex**) in the Printer
-- MIB[1]
-- AND/OR
-- OCTETS: the name of the colorant consumed.

-- ++
-- Time attributes (set by server or device)
--
-- This section of attributes are ones that are set by the
-- server or device that accepts jobs. Two forms of time are
-- provided. Each form is represented in a separate attribute.
-- See section 4.2 on page 18 and section 4.3 on page 19 for the
-- conformance requirements for agents and monitoring
-- applications, respectively. The two forms are:
--
-- **DateAndTime** is an 8 or 11 octet binary encoded year,
-- month, day, hour, minute, second, deci-second with
-- optional offset from UTC. See SNMPv2-TC.
--
-- NOTE: **DateAndTime** is not printable characters; it is
-- binary.
--
-- **JmTimeStampTC** is the time of day measured in the number of
-- seconds since the system was booted. See page 26.
-- ++

jobSubmissionToServerTime(190), -- **JmTimeStampTC** (pg 26)
-- AND/OR

-- **DateAndTime** (SNMPv2-TC)
-- INTEGER: Configuration 2 and 3: The time
-- AND/OR
-- OCTETS: the date and time that the job was submitted to the server.

jobSubmissionToDeviceTime(191), -- **JmTimeStampTC** (pg 26)
-- AND/OR
-- **DateAndTime** (SNMPv2-TC)
-- INTEGER: Configuration 1 and 3: The time
-- AND/OR
-- OCTETS: the date and time that the job was submitted to the device.

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

jobStartedBeingHeldTimeStamp(193), -- **JmTimeStampTC** (pg 26)
-- INTEGER: The time that the job started being held, i.e., the time that the job entered
-- the **pendingHeld** state most recently. 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(194), -- **JmTimeStampTC** (pg 26)
-- AND/OR
-- **DateAndTime** (SNMPv2-TC)
-- INTEGER: The time
-- AND/OR
-- OCTETS: the date and time that the job started processing.

timeSinceStartedProcessing(195), -- **Integer32(-2..2147483647)**
-- INTEGER: The time in milliseconds since the job started processing.

jobCompletedTime(196), -- **JmTimeStampTC** (pg 26)
-- AND/OR
-- **DateAndTime** (SNMPv2-TC)
-- INTEGER: The time
-- AND/OR
-- OCTETS: the date and time that the job completed processing and the medium is
-- completely stacked in the output bin, i.e., when the job entered the **completed**,
-- **canceled**, or **aborted** state.

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

jobProcessingCPUTime(198) -- **Integer32(-2..2147483647)**
-- INTEGER: The amount of CPU time that the job has been processing in seconds,
-- i.e., in the **processing** job state. If the device stops and/or the job enters the
-- **processingStopped** state, that elapsed time SHALL not be included. In other words,

```
            --  the jobProcessingCPUTime value SHOULD be relatively repeatable when the same
            --  job is submitted again.

    }


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 document production instructions that are not one of the identified
            types.

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

        print 0x4
            The job contains some document production instructions that specify printing

        scan  0x8
            The job contains some document production instructions that specify scanning

        faxIn 0x10
            The job contains some document production instructions that specify receive fax
```

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

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

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

mailList 0x100
The job contains some document production 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 7.1.2 on page 21."

SYNTAX **INTEGER(0..2147483647)** -- 31 bits, all but sign bit

JmJobStateReasons1TC ::= TEXTUAL-CONVENTION
STATUS current
DESCRIPTION
"This textual-convention is used with the **jmJobStateReasons1** object to provides additional information regarding the **jmJobState** object values.

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

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

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

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

unknown **0x2**

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

jobIncoming **0x4**

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

jobOutgoing **0x8**

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

jobHoldSpecified **0x10**

The value of the job's **jobHold(52)** attribute (see page 41) is **TRUE**, either set when the job was created or subsequently by an explicit modify job operation. The job SHALL NOT be a candidate for processing until this reason is removed and there are no other reasons to hold the job.

jobHoldUntilSpecified **0x20**

The value of the job's **jobHoldUntil(53)** (see page 41) attribute specifies a time period that is still in the future, either set when the job was created or subsequently by an explicit modify job operation. The job SHALL NOT be a candidate for processing until this reason is removed and there are no other reasons to hold the job.

jobProcessAfterSpecified **0x40**

The value of the job's **jobProcessAfterDateAndTime(51)** (see page 40) attribute specifies a time that is still in the future, either set when the job was created or subsequently by an explicit modify job operation. The job SHALL NOT be a candidate for processing until this reason is removed and there are no other reasons to hold the job.

resourcesAreNotReady **0x80**

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 **0x100**

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 **0x200**

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

jobPrinting **0x400**

The output device is marking media. This attribute is useful for servers and output devices which spend a great deal of time processing when no marking is happening and then want to show that marking is now happening.

jobCanceledByUser **0x800**

The job was canceled by the user, i.e., by a user whose name is the same as the value of the job's **jobOwner** attribute.

jobCanceledByOperator **0x1000**

The job was canceled by the operator, i.e., by a user whose name is different than the value of the job's **jobOwner** attribute.

abortedBySystem **0x2000**

The job was aborted by the system. NOTE - this reason is needed only when the job is not placed in the **aborted** job state.

jobCompletedSuccessfully **0x4000**

The job completed successfully.

jobCompletedWithWarnings **0x8000**

The job completed with warnings.

jobCompletedWithErrors **0x10000**

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[2] and other job submission protocols:

jobPaused **0x20000**

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 **0x40000**

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 **0x80000**

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 resubmit the job (or a copy of the job). When a client could no longer 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 7.1.2 on page 21. The remaining bits are reserved for future standardization and/or registration."

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 the description under **JmJobStateReasons1TC** on page 49.

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** 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.

submissionInterrupted **0x10**

Indicates that the job was not completely submitted for the following reasons: (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. Whether the server or device puts the job into the **pendingHeld** or **aborted** state depends on implementation.

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. Normally, the job is resumed by means outside the job submission protocol, such as some local function on the device.

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. Normally, the job is resumed by means outside the job submission protocol, such as some local function on the device, whereupon the job SHALL transition immediately to the **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 human intervention did not occur 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 device.

queuedInDevice **0x4000**

The job has been queued in a down stream server or device.

jobCleanup **0x8000**

The server/device is performing cleanup activity as part of ending normal processing.

processingToStopPoint **0x10000**

The requester has issued an operation to interrupt the job and the server/device is processing up until the specified stop point occurs.

jobPasswordWait **0x20000**

The server/device has selected the job to be next to process, but instead of assigning resources and started the job processing, the server/device has transitioned the job to the **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.

serviceOffLine **0x400000**

The service/document transform is off-line and accepting no jobs. All **pending** jobs are put into the **pendingHeld** state. This could be true if its input is impaired or broken.

noSpaceOnServer **0x800000**

There is no room on the server to store all of the job. For example, there is no room for the document data.

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**.** The requester SHALL either (1) enter a pin locally at the device or issue a remote operation supplying the PIN in order for the job to be able to proceed.

exceededAccountLimit **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.

An overdraft mechanism SHOULD be included to be user-friendly, so as to minimize the chances that the job cannot finish or that media is wasted. For example, the server/device SHOULD finish the current copy for a job with collated document copies, rather than stopping in the middle of the current document copy.

heldForRetry **0x4000000**

The job encountered some errors that the server/device could not recover from with its normal retry procedures, but the error is worth trying the job later, such as phone number busy or remote file system in-accessible. For such a situation, the server/device SHALL 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. Whether the job is placed in the **pendingHeld** or **aborted** state, depends on implementation.

deviceUnavailable **0x10000000**

This job was aborted by the system because the device is currently unable to accept jobs. Whether the job is placed in the **pendingHeld** or **aborted** state, depends on implementation.

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. Whether the job is placed in the **pendingHeld** or **aborted** state, depends on implementation.

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 7.1.2 on page 21."

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 the description under **JmJobStateReasons1TC** on page 49.

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 7.1.2 on page 21. The remaining bits are reserved for future standardization and/or registration."

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 the description under **JmJobStateReasons1TC** on page 49.

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

none yet defined.

These bit definitions are the equivalent of a type 2 enum except that combinations of them may be used together. See section 7.1.2 on page 21. These bits are reserved for future standardization and/or registration."

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.

-- Implementation of every object in this group is MANDATORY.
-- See Section 4 entitled 'Conformance Considerations' on page 18.

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 Terminology and Job Model on page 11 for the definition of a job set."
        ::= { jmGeneral 1 }

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

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

JmGeneralEntry ::= SEQUENCE {
        jmGeneralNumberOfActiveJobs             Integer32(0..2147483647),
        jmGeneralOldestActiveJobIndex           Integer32(0..2147483647),
        jmGeneralNewestActiveJobIndex           Integer32(0..2147483647),
        jmGeneralJobPersistence                 Integer32(0..2147483647),
        jmGeneralAttributePersistence           Integer32(0..2147483647),
        jmGeneralJobSetName                     OCTET STRING(SIZE(0..63))
}

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 **JmJobStateTC** on page 31 for the exact specification of the semantics of the job states.

If there are no active jobs, the value of this object SHALL be 0."

::= { jmGeneralEntry 1 }

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.

When a job transitions from one of the 'active' states (**pending**, **processing**, **processingStopped**) to one of the 'in-active' states (**pendingHeld**, **completed**, **canceled**, or **aborted),** with a **jmJobIndex** value that matches this object, the agent SHALL advance (or wrap - see **jmGeneralNewestActiveJobIndex**) the value to the next oldest 'active' job, if any.

On the other hand, when a job transitions from one of the 'in-active' states to one of the 'active' state, the agent SHALL reduce (or wrap) the value of this object, if the job's **jmJobIndex** is smaller than the current value.

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

::= { jmGeneralEntry 2 }

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 a new job is accepted by the server or device that the agent is instrumenting, the agent SHALL assign the next available value to the job's **jmJobIndex** that is used for storing job information in the **jmJobIDTable**, the **jmJobTable**, and the **jmAttributeTable**. If the value would exceed the implementation-defined maximum value for **jmJobIndex**, the agent SHALL set the value back to **1**, i.e., wrap around to the beginning of the job tables.

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 pre-mature re-use of **jmJobIndex** value for a newer job while clients retain the same 'stale' value for an older job.

Each time a new job is accepted by the server or device that the agent is instrumenting 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 'in-active' (**pendingHeld** state), the agent SHALL not change the value of **jmGeneralNewestActiveJobIndex** object.

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**. Whenever a job changes from 'in-active' to 'active' (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 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.

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, when the application exceeds the maximum job index (detected by a no such object status returned from a GetNext operation for the next conceptual row), the application SHALL start over at **1** and continue the GetNext operations to find the rest of the active jobs."

::= { jmGeneralEntry 3 }

jmGeneralJobPersistence OBJECT-TYPE
SYNTAX **Integer32(0..2147483647)**
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"The minimum time in seconds for this instance of the Job Set that an entry will 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. Depending on implementation, the value of this object MAY be either: (1) set by the system administrator by means outside this specification or (2) fixed by the implementation."
::= { jmGeneralEntry 4 }

jmGeneralAttributePersistence OBJECT-TYPE
SYNTAX **Integer32(0..2147483647)**
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"The minimum time in seconds for this instance of the Job Set that an entry will 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. The value of this object MAY be either (1) set by the system administrator by means outside this specification or MAY be (2) fixed by the implementation, depending on implementation.

```
1582
1583            This value SHALL be equal to or less than the value of jmGeneralJobPersistence."
1584        ::= { jmGeneralEntry 5 }
1585
1586    jmGeneralJobSetName OBJECT-TYPE
1587        SYNTAX      OCTET STRING(SIZE(0..63))
1588        MAX-ACCESS  read-only
1589        STATUS      current
1590        DESCRIPTION
1591            "The human readable administratively assigned name of this job set (by means outside of this
1592            MIB).  Typically, this name will be the name of the job queue.  If a server or device has only a
1593            single job set, this object can be the administratively assigned name of the server or device itself.
1594            This name does not need to be unique, though each job set in a single Job Monitoring MIB
1595            SHOULD have distinct names.
1596
1597            NOTE - The purpose of this object is to help the user of the job monitoring application
1598            distinguish between several job sets in implementations that support more than one job set."
1599        ::= { jmGeneralEntry 6 }
1600
1601
1602
1603
1604
1605    -- The Job ID Group (Mandatory)
1606
1607    -- The jmJobIDGroup consists entirely of the jmJobIDTable.
1608    --
1609    -- The two key indexes that are used in other tables to index jobs:
1610    -- jmJobSetIndex and jmJobIndex are materialized in this group.
1611    --
1612    -- Implementation of every object in this group is MANDATORY.
1613    -- See Section 4 entitled 'Conformance Considerations' on page 18.
1614
1615    jmJobID  OBJECT IDENTIFIER ::= { jobmonMIBObjects 2 }
1616
1617    jmJobIDTable  OBJECT-TYPE
1618        SYNTAX      SEQUENCE OF JmJobIDEntry
1619        MAX-ACCESS  not-accessible
1620        STATUS      current
1621        DESCRIPTION
1622            "The jmJobIDTable provides a correspondence map (1) between the job submission ID that a
1623            client uses to refer to a job and (2) the jmJobSetIndex and jmJobIndex that the Job
1624            Monitoring MIB agent assigned to the job and that are used to access the job in all of the other
1625            tables in the MIB.  If a monitoring application already knows the jmJobIndex of the job it is
1626            querying, that application NEED NOT use the jmJobIDTable."
1627        ::= { jmJobID 1 }
1628
1629    jmJobIDEntry  OBJECT-TYPE
1630        SYNTAX      JmJobIDEntry
```

```
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
            "The map from (1) the jmJobSubmissionID to (2) the jmJobSetIndex and jmJobIndex.

            An entry SHALL exist in this table for each job, no matter what the state of the job and no
            matter what job set the job is in.  Each job SHALL appear in one and only one job set.

            NOTE - an IMPLICIT statement is NOT provided in the following INDEX clause, since it was
            not an SMIv1 feature.  Therefore, the extra ASN.1 tag SHALL be included in the varbind in the
            SNMP request and the response."
        INDEX  { jmJobSubmissionID }
        ::= { jmJobIDTable 1 }

JmJobIDEntry ::= SEQUENCE {
        jmJobSubmissionID                OCTET STRING(SIZE(1..32)),
        jmJobSetIndex                    Integer32(1..32767),
        jmJobIndex                       Integer32(1..2147483647)
}

jmJobSubmissionID OBJECT-TYPE
        SYNTAX      OCTET STRING(SIZE(1..32))
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
            "A quasi-unique 32-octet string ID which identifies the job uniquely within a particular client-
            server environment.  Either the client or the server assigns the job submission ID for each job.
            The monitoring application whether in the client or running separately, uses the job submission
            ID to help the user identify which jmJobIndex was assigned by the agent.

            There are multiple formats for the jmJobSubmissionID.  Each format SHALL be registered
            using the procedures of a type 2 enum.  See section entitled: 'IANA Registration of enums' on
            page 21.

            The value of jmJobSubmissionID SHOULD be one of the registered format types.  The first
            octet of the string SHALL indicate which registered format is being used.  The ASCII characters
            '0-9', 'A-Z', and 'a-z' will be assigned in order giving 62 possible formats.  The agent SHALL
            assign a string of registered format (0) for any job without a Job Submission ID.

            The format values registered so far are:

              Format
              Number   Description
              ------   ------------
              0        Set by the agent when neither the client nor the
                       server assigned a job submission ID.

              1        octets 3-10:  8-decimal-digit random number
                       octets 11-32: last 22 bytes of the jobName attribute
```

2 octets 3-10: 8-decimal-digit sequential number
octets 11-32: Client MAC address

3 octets 3-10: 8-decimal-digit sequential number
octets 11-32: last 22 bytes of the client URL

.. to be registered according to procedures of a type 2 enum. See section 7.3 on page 22.

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 a very low number, but is not intended to be an absolute guarantee of uniqueness. None-the-less, collisions could occur, but without bad consequences, since this MIB is intended to be used only for monitoring jobs, not for controlling and managing them."

::= { jmJobIDEntry 1 }

jmJobSetIndex OBJECT-TYPE
SYNTAX **Integer32(1..32767)**
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"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 **jmJobIndex** value permits the management application to access the other tables to obtain the job-specific objects. This value SHALL be the same for a job in the **jmJobIDTable** as the corresponding **jmJobSetIndex** value in the **jmJobTable** and **jmAttributeTable** for this job.

The value(s) of the **jmJobSetIndex** SHALL be persistent across power cycles, so that clients that have retained **jmJobSetIndex** 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 Terminology and Job Model on page 11 for the definition of a job set."
::= { jmJobIDEntry 2 }

jmJobIndex OBJECT-TYPE
SYNTAX **Integer32(1..2147483647)**
MAX-ACCESS read-only
STATUS current
DESCRIPTION

```
1725            "The sequential, monatonically increasing identifier index for the job generated by the server or
1726            device when that server or device accepted the job.  This index value permits the management
1727            application to access the other tables to obtain the job-specific row entries.  This value SHALL
1728            be the index used in the jmJobTable and jmAttributeTable for this job.
1729
1730            See jmGeneralNewestActiveJobIndex on page 58 for a discussion about the largest value of
1731            jmJobIndex for an implementation.
1732
1733            Agents instrumenting systems that contain jobs with a job identifier of 0 SHALL map the job
1734            identifier value 0 to a jmJobIndex value that is one higher than the highest job identifier value
1735            that any job can have on that system."
1736        ::= { jmJobIDEntry 3 }
1737
1738
1739
1740
1741  -- The Job Group (Mandatory)
1742
1743  -- The jmJobGroup consists entirely of the jmJobTable.
1744  --
1745  -- Implementation of every object in this group is MANDATORY.
1746  -- See Section 4 entitled 'Conformance Considerations' on page 18.
1747
1748  jmJob  OBJECT IDENTIFIER ::= { jobmonMIBObjects 3 }
1749
1750  jmJobTable  OBJECT-TYPE
1751        SYNTAX      SEQUENCE OF JmJobEntry
1752        MAX-ACCESS  not-accessible
1753        STATUS      current
1754        DESCRIPTION
1755            "The jmJobTable consists of basic job state and status information for each job in a job set that
1756            (1) monitoring applications need to be able to access in a single SNMP Get operation, (2) that
1757            have a single value per job, and (3) that SHALL always be implemented."
1758        ::= { jmJob 1 }
1759
1760  jmJobEntry  OBJECT-TYPE
1761        SYNTAX      JmJobEntry
1762        MAX-ACCESS  not-accessible
1763        STATUS      current
1764        DESCRIPTION
1765            "Basic per-job state and status information.
1766
1767            An entry SHALL exist in this table for each job, no matter what the state of the job is.  Each job
1768            SHALL appear in one and only one job set."
1769        INDEX  { jmJobSetIndex, jmJobIndex }
1770        ::= { jmJobTable 1 }
1771
1772  JmJobEntry ::= SEQUENCE {
        jmJobState                                  JmJobStateTC,       -- pg 31
```

```
        jmJobStateReasons1                  JmJobStateReasons1TC,  -- pg 49
        jmNumberOfInterveningJobs           Integer32(-2..2147483647),
        jmJobKOctetsRequested               Integer32(-2..2147483647),
        jmJobKOctetsProcessed               Integer32(-2..2147483647),
        jmJobImpressionsRequested           Integer32(-2..2147483647),
        jmJobImpressionsCompleted           Integer32(-2..2147483647)
}


jmJobState OBJECT-TYPE
    SYNTAX      JmJobStateTC            -- See page 31
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The current state of the job (pending, processing, completed, etc.). Even though the
        JmJobStateTC textual-convention defines nine values for job states, agents SHALL only
        implement those states which are appropriate for the particular implementation. In other words,
        all possible enums for this object SHALL be reported if implemented by the device and available
        to the agent. 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 keep a job in the completed, canceled, or
        aborted state before removing the job from the jmJobIDTable and jmJobTable is specified by
        the value of the jmGeneralJobPersistence object."
    ::= { jmJobEntry 1 }

jmJobStateReasons1 OBJECT-TYPE
    SYNTAX      JmJobStateReasons1TC        -- See page 49
    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.

        NOTE - The jobStateReasonsn (n=2..4) attributes (see page 36) provide further additional
        information about the job's current state.

        Implementation of these values is OPTIONAL, i.e., an agent NEED NOT implement them, even
        if (1) the device supports the functionality represented by the reason and (2) is available to the
        agent. These values MAY be used with any job state or states for which the reason makes
        sense. Furthermore, when implemented, the agent SHALL return these values when the reason
        applies and SHALL NOT return them when the reason no longer applies whether the value of
        the job's jmJobState object changed or not. When the job does not have any reasons for being
        in its current state, the agent SHALL set the value of the jmJobStateReasons1 object and
        jobStateReasonsn attributes to 0.
```

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."

::= { jmJobEntry 2 }

jmNumberOfInterveningJobs OBJECT-TYPE
SYNTAX **Integer32(-2..2147483647)**
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"The number of jobs that are expected to be processed *before* this job is processed according to the implementation's queuing algorithm if no other jobs were to be submitted. In other words, this value is the job's queue position. The agent SHALL return a value of **0** for this attribute when this job starts processing (since there are no jobs in front of the job)."
::= { jmJobEntry 3 }

jmJobKOctetsRequested 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 SHALL be represented as **0**, 1-1024 octets SHALL be represented as **1**, 1025-2048 SHALL be represented as **2**, etc.

The server/device MAY update the value of this attribute after each document has been transferred to the server/device or the server/device MAY provide this value after all documents have been transferred to the server/device, depending on implementation. In other words, while the job is in the **pendingHeld** state with the **jmJobStateReasons1** object containing a **jobIncoming** value, the value of the **jmJobKOctetsRequested** object depends on implementation and MAY not correctly reflect the size of the job.

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."
::= { jmJobEntry 4 }

jmJobKOctetsProcessed OBJECT-TYPE
SYNTAX **Integer32(-2..2147483647)**
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"The current number of octets processed by the server or device measured in units of K (1024) octets. The agent SHALL round the actual number of octets processed up to the next higher K.

Thus 0 octets SHALL be represented as **0**, 1-1024 octets SHALL be represented as **1**, 1025-2048 octets SHALL be **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 makes only a single pass over the document, the final value SHALL be equal to the value of the **jmJobKOctetsRequested** object. For implementations where multiple copies are produced by the interpreter making multiple passes over the document, the final value SHALL be a multiple of the value of the **jmJobKOctetsRequested** object.

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

NOTE - The **jmJobKOctetsProcessed** object can be used in the numerator with the **jmJobKOctetsRequested** object in the denominator in order to produce a "thermometer" that indicates the progress of the job, provided that the multiplicative factor is taken into account for some implementations of multiple copies."

::= { jmJobEntry 5 }

jmJobImpressionsRequested OBJECT-TYPE
SYNTAX **Integer32(-2..2147483647)**
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"The number of impressions requested by this job to produce."
::= { jmJobEntry 6 }

jmJobImpressionsCompleted OBJECT-TYPE
SYNTAX **Integer32(-2..2147483647)**
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"The current 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."
::= { jmJobEntry 7 }

-- The Attribute Group (Mandatory)

-- The **jmAttributeGroup** consists entirely of the **jmAttributeTable.**
--
-- Implementation of the two objects in this group is MANDATORY.
-- See Section 4 entitled '**Conformance Considerations**' on page 18.
--
-- A few attributes are MANDATORY for agent conformance, and the rest
-- are OPTIONAL. See the specification of the **JmAttributeTypeTC** on

```
-- page 33 for which attributes are MANDATORY for agents to implement.

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."
        ::= { 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 JmAttributeTypeTC on page 33).  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 jmJobTable for at least the minimum time after a
                job completes as specified by the jmGeneralAttributePersistence (see page 59).

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

JmAttributeEntry ::= SEQUENCE {
        jmAttributeTypeIndex            JmAttributeTypeTC,      -- pg 33
        jmAttributeInstanceIndex        Integer32(1..32767),
```

```
        jmAttributeValueAsInteger       Integer32(-2..2147483647),
        jmAttributeValueAsOctets        OCTET STRING(SIZE(0..63))
}

jmAttributeTypeIndex OBJECT-TYPE
    SYNTAX      JmAttributeTypeTC       -- See page 33
    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 and document attributes include:  jobCopiesRequested,
        documentCopiesRequested, jobCopiesCompleted, documentCopiesCompleted, fileName,
        and documentName.

        Examples of required and consumed resource attributes include:  pagesRequested,
        pagesCompleted, mediumRequested, and mediumConsumed, 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 or documentFormat, 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 definition (see page 33) has the tag:
        'INTEGER:'.

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

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

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

Since the agent MAY add attributes representing resources to this table while the job is waiting to be processed or being processed, which can be a long time before any of the resources are actually used, the agent SHALL set the value of the **jmAttributeValueAsInteger** object to **0** for resources that the job has not yet consumed.

Attributes for which the concept of an integer value is meaningless, such as **fileName**, **interpreter,** and **physicalDevice,** do *not* have the 'INTEGER:' tag in the **JmAttributeTypeTC** definition and so SHALL return a value of (**-1**) to indicate **other** for **jmAttributeValueAsInteger**.

For attributes which do have the 'INTEGER:' tag in the **JmAttributeTypeTC** definition, if the integer value is not (yet) known, the value SHALL be (**-2**) to represent unknown counting integers, **(2)** to represent unknown enum values, or the attribute row SHALL not be present in the table."

::= { 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** definition (see page 33) has the tag: 'OCTETS:'.

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

Attributes for which the concept of an octet string value is meaningless, such as **pagesCompleted**, do *not* have the tag 'OCTETS:' in the **JmAttributeTypeTC** definition and so the agent SHALL return a zero length string for the value of the **jmAttributeValueAsOctets** object."

::= { jmAttributeEntry 4 }

```
-- 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 }

                -- OBJECT jmAttributeTypeIndex
                -- SYNTAX      INTEGER {
                --    jobOwner(20)
                -- }
        -- DESCRIPTION
                --"It is conformant for an agent to implement the one mandatory
                -- attribute.  Any additional attributes are OPTIONAL,
                -- i.e., an agent NEED NOT represent any additional
                -- attributes that the server or device implements.  However, a
                -- client SHALL accept all of the attributes from an agent and
                -- either display them to its user or ignore them.
                --
                -- NOTE - SMI does not allow an enum to be declared as mandatory
                -- if that enum is not a member of a group, but
                -- jmAttributeTypeIndex cannot be a member of a group and still
                -- be not-accessible.  So this MIB spec comments the MANDATORY
                -- attributes as if SMI allowed such a declaration in order to
                -- declare the MANDATORY attributes."

-- There are no CONDITIONALLY MANDATORY or OPTIONAL groups.

        ::= { jmMIBConformance 1 }

jmMIBGroups     OBJECT IDENTIFIER ::= { jmMIBConformance 2 }

jmGeneralGroup OBJECT-GROUP
        OBJECTS {
                jmGeneralNumberOfActiveJobs, jmGeneralOldestActiveJobIndex,
```

```
2098                **jmGeneralNewestActiveJobIndex, jmGeneralJobPersistence,**
2099                **jmGeneralAttributePersistence, jmGeneralJobSetName**}
2100            STATUS  current
2101            DESCRIPTION
2102                "The general group."
2103            ::= { jmMIBGroups 1 }
2104
2105    jmJobIDGroup OBJECT-GROUP
2106            OBJECTS {
2107                **jmJobSetIndex**, **jmJobIndex** }
2108            STATUS  current
2109            DESCRIPTION
2110                "The job ID group."
2111            ::= { jmMIBGroups 2 }
2112
2113    jmJobGroup OBJECT-GROUP
2114            OBJECTS {
2115                **jmJobState**, **jmJobStateReasons1, jmNumberOfInterveningJobs,**
2116                **jmJobKOctetsRequested, jmJobKOctetsProcessed, jmJobImpressionsRequested,**
2117                **jmJobImpressionsCompleted** }
2118            STATUS  current
2119            DESCRIPTION
2120                "The job group."
2121            ::= { jmMIBGroups 3 }
2122
2123    jmAttributeGroup OBJECT-GROUP
2124            OBJECTS {
2125                **jmAttributeValueAsInteger, jmAttributeValueAsOctets** }
2126            STATUS  current
2127            DESCRIPTION
2128                "The attribute group."
2129            ::= { jmMIBGroups 4 }
2130
2131
2132    END
```

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

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

The job model does not specify how accounting and auditing is implemented, except to 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 logs 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.

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

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

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

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

```
SUBMISSIONID = "id string"
```

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

The entire PJL JOB command with the optional parameter would be of the form:

```
@PJL JOB SUBMISSIONID = "id string"
```

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

14. Bibliography

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

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

[3] Internet Printing Protocol (IPP), in progress on the IETF standards track. See **draft-ietf-ipp-model-01.txt**. See also **http://www.pwg.org/ipp/index.html**

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

[5] MIB-II, RFC 1213.

[6] Host Resources MIB, RFC 1514

[7] RFC 2119

15. Author's Addresses

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

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

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

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

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

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 comments to the printmib WG using the Job Monitoring Project (JMP)
Mailing List: jmp@pwg.org

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

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 Ferrel - Canon
Steve Gebert - IBM
Robert Herriot - Sun Microsystems Inc.
Shige Kanemitsu - Kyocera
David Kellerman - Northlake Software
Rick Landau - Digital
Harry Lewis - IBM
Pete Loya - HP
Ray Lutz - Cognisys
Jay Martin - Underscore
Mike MacKay, Novell, Inc.
Stan McConnell - Xerox
Carl-Uno Manros, Xerox, Corp.
Pat Nogay - IBM
Bob Pentecost - HP
Rob Rhoads - Intel
David Roach - Unisys
Hiroyuki Sato - Canon
Bob Setterbo - Adobe
Gail Songer, EFI
Mike Timperman - Lexmark

2277 Randy Turner - Sharp
2278 William Wagner - Digital Products
2279 Jim Walker - Dazel
2280 Chris Wellens - Interworking Labs
2281 Rob Whittle - Novell
2282 Don Wright - Lexmark
2283 Lloyd Young - Lexmark
2284 Atsushi Yuki - Kyocera
2285 Peter Zehler, Xerox, Corp.

16. 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.