Job Monitoring MIB

From: Tom Hastings
Date: 04/04/97
Version: 0.8
File: ftp://ftp.pwg.org/pub/jmp/mibs/jmp-mib.doc .pdf jmp-mibr.doc .pdf .pdr

Status: Fourth draft MIB that corresponds to the changes agreed to at the JMP meeting, 04/04/97 in Austin. Harry Lewis's changes to eliminate the Queue and Completed tables and to replace the Job table with the Job ID and Job State table have been incorporated. See the change history. The Internet-Draft was not posted in time and with these changes, we will not present any MIB document at the IETF meeting on 04/08/97 in Memphis. Instead we will present slides on the current status explaining the tables, which are: General, Job ID, Job State, and Attributes.

The MIB has been greatly simplified so that now there are only 13 objects in the MIB. There are 57 attributes, of which only 7 are mandatory.

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

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

The greatly simplified specifications of each object is derived from the ISO DPA attribute specifications in most cases. I've moved the full ISO DPA specifications to a separate document. I've indicated ISSUES in a separate document that we have identified as issues but have not resolved. 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.

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

INTERNET-DRAFT

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

Job Monitoring MIB - V0.7

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

Expires Oct 4, 1997

Status of this Memo

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

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

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

Abstract

This Internet-Draft specifies a set of 13 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

196

Job Monitoring MIB

1. Introduction

The Job Monitoring MIB consists of a 5-object General Group, a 2-object Job Submission ID Group, a 4-object Job State 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 State and Attribute tables. The Job State table contains the job state and copies of three salient attributes for each job's current state. 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. The Job Monitoring MIB is intended to be instrumented by an agent within a printer or the first server closest to the printer, where the printer is either directly connected to the server only or the printer does not contain the job monitoring MIB agent. It is recommended that implementations place the SNMP agent as close as possible to the processing of the print job. This MIB applies to printers with and without spooling capabilities. This MIB is designed to be compatible with most current commonly-used job submission protocols. In most environments that support high function job submission/job control protocols, like ISO DPA, those protocols would be used to monitor and manage print jobs rather than using the Job Monitoring MIB.

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 (Refer to RFC 1759, Appendix D - Roles of Users).

User:

- Provide the ability to identify the least busy printer. The user will be able to determine the number and size of jobs waiting for each printer. No attempt is made to actually predict the length of time that jobs will take.
- Provide the ability to identify the current status of the job (user queries).
- Provide a timely notification that the job has completed and where it can be found.
- Provide error and diagnostic information for jobs that did not successfully complete.

Operator:

- Provide a presentation of the state of all the jobs in the print system.
- Provide the ability to identify the user that submitted the print job.
- Provide the ability to identify the resources required by each job.

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

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

Capacity Planner:

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

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

Accountant:

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

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

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

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 of the jobs in a queue, which is generalized 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 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 **canceled** and **completed** 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** or **canceled**. Instead, the application shall query each job that the application's shadow copy shows was not **complete** or **canceled** 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, so that coherence is maintained between the two protocols and information presented to end users and system operators. However, the job monitoring MIB is intended to be used with printers that implement other job submitting and management protocols, such as IEEE 1284.1 (TIPSI), as well as with ones that do implement ISO DPA. So nothing in the job monitoring MIB shall require implementation of the ISO DPA protocol.

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

2. Terminology and Job Model

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

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

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

A *job set* is a set of jobs that are queued and scheduled together according to a specified scheduling algorithm for a specified device or set of devices. For implementations that embed the SNMP agent in the device, the MIB job set normally represents *all* the jobs known to the device, 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.

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 for which the Job Monitoring MIB is intended to be used. The diagram in the Printer MIB entitled: "One Printer's View of the Network"[1] is assumed for this MIB as well. Please refer to that diagram to aid in understand the following system configurations. To simplify the pictures, the *devices* are shown as *printers*. See Goals section.

3.1 Configuration 1 - client-printer

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

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

The job submitting **client** and/or **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.

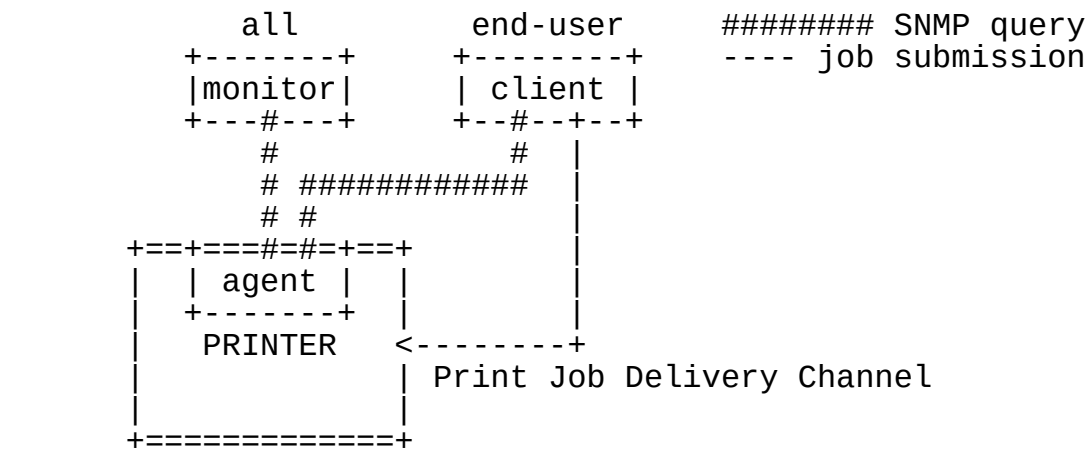

Figure 1 - Configuration 1 - client-printer - agent in the printer

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

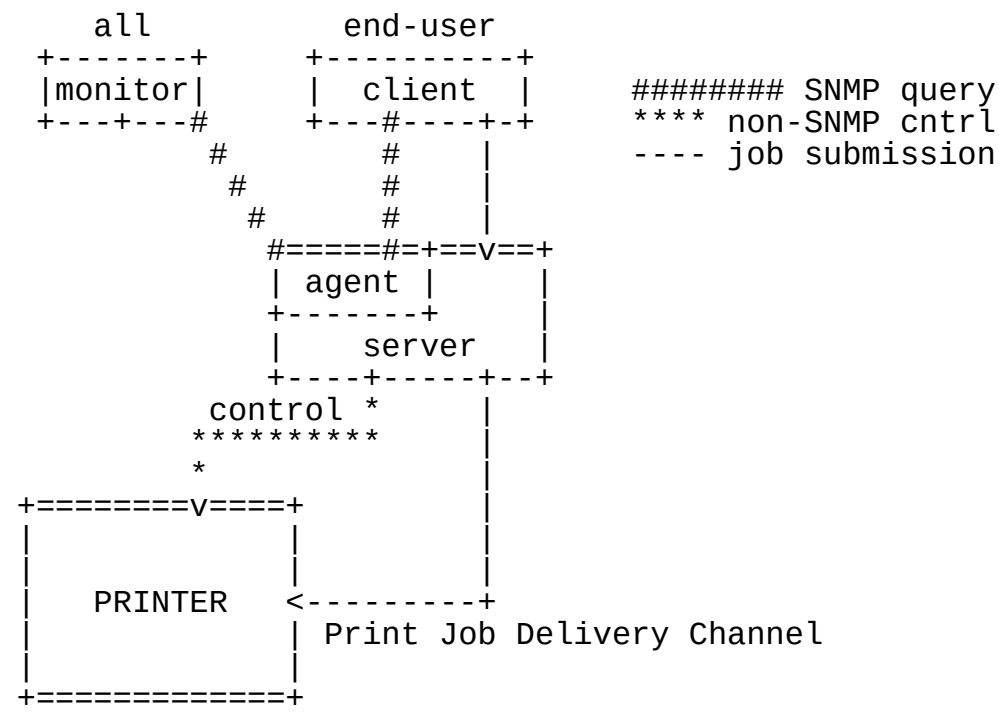

Figure 2 - Configuration 2 - client-server-printer - agent in the server

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

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.

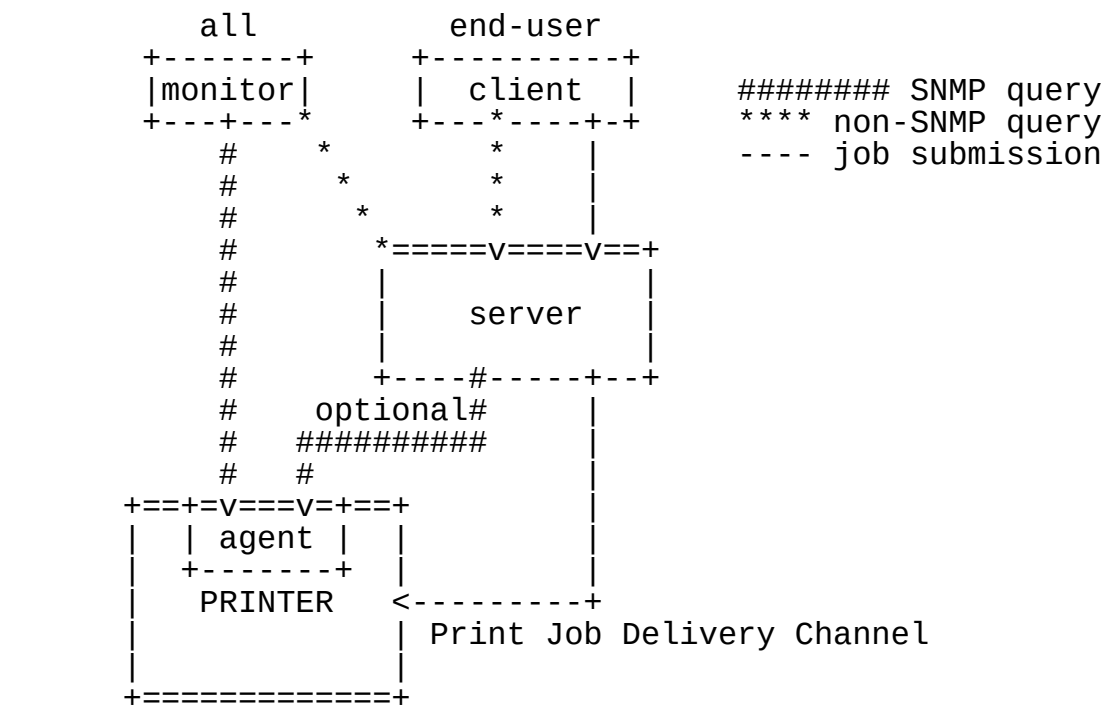

Figure 3 - Configuration 3 - client-server-printer - client monitors printer agent and server

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

1. Multiple **clients** may submit jobs to a **server**.
2. Multiple **clients** may monitor a **server**.
3. Multiple **monitors** may monitor a **server**.

4. A **client** may submit jobs to multiple **servers**.
5. A **monitor** may monitor multiple **servers**.
6. Multiple **servers** may submit jobs to a **printer**.
7. Multiple **servers** may control a **printer**.

4. Conformance Considerations

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

4.1 Conformance Terminology

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

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

4.2 Agent Conformance Requirements

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

4.2.1 MIB II System Group objects

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

4.2.2 MIB II Interface Group objects

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

4.2.3 Printer MIB objects

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

4.3 Job Monitoring Application Conformance Requirements

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

A job monitoring application shall accept all enum values and bit vector bits specified in this standard and additional ones that may be registered with IANA and shall either present them to the user or ignore them. See Section 7 entitled "IANA Considerations" on page 18.

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 57.

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

6. Internationalization Considerations

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

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

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

7. IANA Considerations

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

7.1 IANA Registration of enums

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

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

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

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

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

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

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

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

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

7.2 IANA Registration of bit string values

This draft contains the following bit string textual-conventions:

1. **JmJobStateReasonsTC**

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

8. Security Considerations

8.1 Read-Write objects

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

8.2 Read-Only Objects In Other User's Jobs

The security policy of some sites may be that unprivileged users can only get the objects from jobs that they submitted, plus a few minimal objects from other jobs, such as the **jobKOctetsRequested** and **jobKOctetsCompleted** attributes**,** so that a user can tell how busy a printer is. Other sites might allow all unprivileged users to see all objects of all jobs. It is up to the agent to implement any such restrictions based on the identification of the user making the SNMP request. This MIB does not require, nor does it specify how, such restrictions would be implemented. A monitoring application should enforce the site security policy with respect to returning information to an unprivileged end user that is using the monitoring application to monitor jobs that do not belong to that user, i.e., the **jobOwner** attribute in the **jmAttributeTable** does not match the user's user name. See the **JmAttributeTypeTC** textual convention on page 38 and the **jmAttributeTable**.

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

9. Returning Objects With No Value In Mandatory Groups

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

10. Notification and Traps

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

11. MIB specification

The following pages constitute the actual Job Monitoring MIB.

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

IMPORTS
    MODULE-IDENTITY, OBJECT-TYPE, experimental,
    Integer32                                          FROM SNMPv2-SMI
    TEXTUAL-CONVENTION                                 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

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

jobmonmib MODULE-IDENTITY
    LAST-UPDATED "9704040000Z"
    ORGANIZATION "IETF Printer MIB Working Group"
    CONTACT-INFO
        "Tom Hastings
        Postal:  Xerox Corp.
                 Mail stop ESAE-231
                 701 S. Aviation Blvd.
                 El Segundo, CA 90245

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

        File: jmp-mib.doc,  .pdf,  .txt,  .mib
        Version: 0.8"
    ::= { temp 105 }



-- Textual conventions for this MIB module

```

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

        NOTE - JmTimeTC 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 MIB-II.)

        NOTE - JmTimeTC 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 65).  The TimeStamp
        textual-convention defined in SMIv2 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)



JmTimeIntervalTC ::= TEXTUAL-CONVENTION
    STATUS      current
    DESCRIPTION
        "A period of time, measured in units of seconds.

        NOTE - JmTimeIntervalTC is defined in the same units as
        JmTimeTC, namely seconds.

        NOTE - JmTimeIntervalTC is defined as an Integer32 so that it
        can be used as a value of an attribute which is represented as
        the value of the jmAttributeValueAsInteger object (see page 65).
        The TimeIntervalTC textual-convention defined in SNMP-TC is
        defined as an Integer32, so it could be used in this MIB, except
        that TimeIntervalTC is defined in 100ths of a second, not in
        units of seconds."
    SYNTAX      INTEGER(0..2147483647)



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

        Management applications shall be prepared to receive all the
        standard job states.  Servers and devices are not required to
        generate all job states, only those which are appropriate for
        the particular implementation.  However, the following states
        are mandatory for a server or device implementation:
```

```
788
789             processing(5)
790             needsAttention(7)
791             canceled(8)
792             completed(9)
793
794         See Section 12 entitled 'Job Life Cycle' on page 69 for
795         additional job state semantics, legal job state transitions, and
796         implementation considerations.
797
798         A companion textual convention (JmJobStateReasonsTC) and
799         corresponding attribute (jobStateReasons) provide additional
800         information about job states.  While the job states cannot be
801         added to without impacting deployed clients, it is the intent
802         that additional JmJobStateReasonsTC enums can be defined without
803         impacting deployed clients.  In other words, the
804         JmJobStateReasonsTC is intended to be extensible.  See page 43.
805
806         The following job state standard values are defined:"
807
808       -- This is a type 2 enumeration.  See Section 7.1 on page 18.
809       SYNTAX      INTEGER {
            other(1),           -- The job state is not one of the defined
                                -- states.

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

            held(3),            -- The job is not yet a candidate for
                                -- processing for any number of reasons.
                                -- The reasons are represented as bits in
                                -- the jobStateReasons attribute.  Some
                                -- reasons are used in other states to give
                                -- added information about the job state.
                                -- See the JmJobStateReasonsTC textual
                                -- convention for the specification of each
                                -- reason and in which states the reasons
                                -- may be used.

            pending(4),         -- The job is a candidate for processing,
                                -- but is not yet processing.

            processing(5),      -- The job is using one or more document
                                -- transforms which include purely software
                                -- processes, such as interpreting a PDL,
                                -- and hardware devices, but is not yet
                                -- making marks on a medium.
                                --
                                -- If an implementation does not distinguish
                                -- between processing and printing, then the
                                -- processing state shall be implemented.

            printing(6)         -- The job is printing, i.e., making marks
```

```
                        -- on a medium.
                        --
                        -- If an implementation does not distinguish
                        -- between processing and printing, then the
                        -- processing state shall be implemented.

    needsAttention(7),  -- The job is using one or more devices, but
                        -- has encountered a problem with at least
                        -- one device that requires human
                        -- intervention before the job can continue
                        -- using that device.  Examples include
                        -- running out of paper or a paper jam.
                        --
                        -- Usually devices indicate their condition
                        -- in human readable form locally at the
                        -- device.  The management application can
                        -- obtain more complete device status
                        -- remotely by querying the appropriate
                        -- device MIB using the job's jmDeviceIndex
                        -- object in the Job Monitoring MIB.

    canceled(8),        -- The job is in the process of being
                        -- terminated by the server or device or has
                        -- completed terminating the job, either
                        -- because the client canceled the job or
                        -- because a serious problem was encountered
                        -- by a document transform while processing
                        -- the job.  The job's jobStateReasons
                        -- attribute shall contain the reasons that
                        -- the job was canceled.  The job shall
                        -- remain in the canceled state for the same
                        -- period of time as if the job had
                        -- completed, before transiting to the
                        -- unknown state.  See the completed state
                        -- description.

    completed(9)        -- The job has (1) completed after
                        -- processing/printing and all of the media
                        -- have been successfully stacked in the
                        -- output bin(s).
                        --
                        -- The job has completed successfully or
                        -- with warnings or errors.  The job's
                        -- jobStateReasons attribute shall contain
                        -- the reasons that the job has entered the
                        -- completed state.
                        --
                        -- The length of time that a job may be in
                        -- the completed state, before transitioning
                        -- to unknown, is specified by the value of
                        -- the jmGeneralJobPersistence object.  In
                        -- addition, the agent shall maintain all of
                        -- the attributes in the jmAttributeTable
```

```
                        -- for at least the time specified in the
                        -- jmGeneralAttributePersistence object, so
                        -- that a management application accounting
                        -- program can copy all the attributes to an
                        -- accounting log.
    }


JmAttributeTypeC ::= TEXTUAL-CONVENTION
    STATUS      current
    DESCRIPTION
        "The type of 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 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 otherwise in
        JmAttributeTypeTC, an agent shall ensure that each attribute
        item occurs only once in the jmAttributeTable for a job.
        Attributes that may appear multiple times in the
        jmAttributeTable for a job are indicated in their specification
        in the JmAttributeTypeTC (see page 25).  However, such attribute
        items shall not contain duplicates for 'intensive' (as opposed
        to 'extensive') attributes.

                For example, each documentFormatEnum attribute entry
                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.
```

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

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

An agent shall create a row in the **jmAttributeTable** for each attribute that is (1) supplied with a job when the job is accepted by a server or printer or that (2) the server or printer supplies as a default either when the job is accepted or later during processing. An agent shall not create a row for any attribute that was neither supplied with the job nor supplied by the server or printer as a default.

Some attributes are mandatory for conformance, and the rest are conditionally mandatory. An agent shall instrument any mandatory attribute. If the server or printer does not provide access to the information about the mandatory attribute, the agent shall return the **'unknown'** value. An agent shall instrument any conditionally mandatory attribute if the server or printer provides access to the information about the attribute to the agent. If the server or printer does *not* provide access to the information about the conditionally mandatory attribute, the agent shall *not* create the row in the **jmAttributeTable**.

The mandatory attributes are the ones required to have copies in the **jmJobStateTable**. The mandatory attributes are:

jobState
numberOfInterveningJobs
deviceAlertCode
jobKOctetsRequested
jobKOctetsCompleted
impressionsRequested
impressionsCompleted
outputBinName

The table of contents lists the attributes in order to help see the order of OID assignment which is the order that the GetNext operation returns attributes.

The standard attribute types defined so far are:"

```
909
910       -- This is a type 2 enumeration.  See Section 7.1 on page 18.
911       SYNTAX      INTEGER {
           -- jm            Description - including Octets: or Integer:
           -- Attribute     to specify whether the value is represented
           -- TypeIndex     in the jmAttributeValueAsOctets or the
           --               jmAttributeValueAsInteger object,
           --               respectively.

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


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

           jobState(2)   -- The current state of the job (pending,
                         -- processing, held, etc.)
                         --
                         -- Management applications shall be prepared to
                         -- receive all the standard job states.
                         -- Servers and devices are not required to
                         -- generate all job states, only those which
                         -- are appropriate for the particular
                         -- implementation.
                         --
                         -- A companion textual convention
                         -- (JmJobStateReasonsTC) and corresponding
                         -- attribute (jobStateReasons) provide
                         -- additional information about job states.
                         -- While the job states cannot be added to
                         -- without impacting deployed clients, it is
                         -- the intent that additional
                         -- JmJobStateReasonsTC enums can be defined
                         -- without impacting deployed clients.  In
                         -- other words, the JmJobStateReasonsTC is
                         -- intended to be extensible.  See page 43.
                         --
                         -- This attribute is a type 2 enum.

           jobStateAsso  -- Integer:  The value of the most relevant
           ciatedValue(  -- attribute associated with the job's current
           3)            -- state.
                         --
                         -- Which attribute depends on the job's current
                         -- state (as specified by the value of the
                         -- jmJobState object and the jobState
                         -- attribute) as follows:
                         --
                         -- jmJobState       Associated Attribute   Page
```

```
                -- /jobState
                --
                -- held              jobStartedBeingHeldTime  41
                -- pending           numberOfInterveningJobs  43
                -- processing        jobKOctetsRequested      47
                -- printing          impressionsRequested     49
                -- needsAttention    deviceAlertCode          29
                -- canceled          impressionsCompleted     38
                -- completed         outputBinName            34
                --
                -- NOTE - The jobStateAssociatedValue attribute
                -- selects from amongst seven mandatory
                -- attributes that attribute that is most
                -- relevant to the job's current state.  the
                -- jobStateAssociatedValue attribute is
                -- provided as an efficiency improvement, so
                -- that an application can obtain the most
                -- relevant attribute for each job's current
                -- state (1) without first having to determine
                -- the job's state or (2) having to request all
                -- seven mandatory attributes in the same
                -- GetNext operation that obtains the next job
                -- in the next conceptual row in the
                -- jmAttributeTable.

jobStateReas    -- Octets:  Additional information regarding
ons(4)          -- the jmJobState/jobState object/attribute.
                -- The jobStateReasons attribute identifies the
                -- reason or reasons that the job is in the
                -- held, pending, processing, needsAttention,
                -- canceled, retained, or completed state.  The
                -- server shall indicate the particular
                -- reason(s) by setting the value of the
                -- jobStateReasons attribute.  While the job
                -- states cannot be added to without impacting
                -- deployed clients, it is the intent that
                -- additional JmJobStateReasonsTC enums can be
                -- defined without impacting deployed clients.
                -- In other words, the JmJobStateReasonsTC is
                -- intended to be extensible.  See page 43.
                --
                -- When the job does not have any reasons for
                -- being in its current state, the server shall
                -- set the value of the jobStateReasons
                -- attribute to a bit string containing all
                -- zeros.
                --
                -- Bits in the bit string are assigned starting
                -- with the most significant bit in the most
                -- significant octet which is called bit 1.
                -- Bit 2 is the next most significant bit in
                -- the most significant octet, etc.  Bit 9 is
                -- the most significant bit in the second most
```

```
                -- significant octet, etc., up to the maximum
                -- bit:  504 (= 8 x 63).  See JobStateReasonsTC
                -- on page 43.
                --
                -- An agent only needs to return the most
                -- significant octet up to the least
                -- significant octet that contains a non-zero
                -- bit.  The remaining octets need not be
                -- returned.
                --
                -- If all bits are zero, the agent may return
                -- an OCTET STRING of zero length.
                -- Alternatively, an agent may always return a
                -- fixed number of octets starting with the
                -- most significant octet and running through
                -- the least significant octet that could ever
                -- have a one bit in it for that
                -- implementation.
                --
                -- This attribute is a type 2 bit string.  See
                -- Section 7 entitled 'IANA Considerations' on
                -- page 18.

numberOfInte    -- Integer:  The number of jobs that are
rveningJobs(    -- expected to be processed before this job is
5)              -- 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 object when the
                -- job starts processing (since there are no
                -- jobs in front of the job).

deviceAlertC    -- The device alert code when the job is
ode(6)          -- 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 using the PrtAlertCodeTC
                -- textual convention or equivalent.

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


-- ********************************************************
-- Job Identification attributes
```

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

jobName(8) -- 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.
--
-- 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** resource item of the first document in the job or the **fileName** resource item of the first document in the job.
--
-- The **jobName** attribute is distinguished from the **jobComment** attribute, in that the **jobName** attribute is intended to permit the submitting user to distinguish between different jobs that he/she has submitted. The **jobComment** attribute is intended to be free form additional information that a user might wish to use to communicate with himself/herself, such as a reminder of what to do with the results or to indicate a different set of input parameters were tried in several different job submissions.

jobServiceTypes(9) -- Integer: Specifies the type(s) of service to which the job has been submitted (print, fax, scan, etc.) as defined by the **JmJobServiceTypesTC** on page 42. The service type is represented as a **BITS** datatype that is bit encoded with each job service type so that more general and arbitrary services can be created, such as services with more than one destination type, or ones with only a source or only a destination. For example, a job service might scan, fax, and print a single job. In this case, three bits would be set in the **jobServiceTypes** attribute, corresponding to the values: 8+32+4=44, respectively.

```
                    --
                    -- 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 the object is
                    -- in the job identification category.
                    --
                    -- This attribute is a type 2 enum.

jobOwner(10)        -- 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.

jobAccountNa        -- Octets:  Arbitrary binary information which
me(11),             -- 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.

jmJobDeviceN        -- The administratively defined coded character
ameOrQueueRe        -- set name of the target device or queue.  Its
quested(12)         -- value corresponds to the Printer MIB:
                    -- prtGeneralPrinterName object (added to the
                    -- draft Printer MIB) for printers.  For
                    -- servers, this object is the name that users
                    -- supply to indicate whether they want the job
                    -- to be processed, typically, but not limited
                    -- to, a job queue name or logical printer
                    -- name.

jobSourceCha        -- Integer:  The index of the row in the
nnelIndex(13        -- associated Printer MIB of the channel which
```

```
),              -- is the source of the print job.  See RFC
                -- 1759.
                --
                -- Must be 1 or greater.
                --
                -- NOTE - the Job Monitoring MIB points to the
                -- Channel row in the Printer MIB, so there is
                -- no need for a port object in the Job
                -- Monitoring MIB, since the PWG is adding a
                -- prtChannelInformation object to the Channel
                -- table of the draft Printer MIB.

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

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

fileName(16)    -- Octets: The coded character set file name of
,               -- the document.
                --
                -- A row with this attribute item may appear
                -- more than once in the jmAttributeTable for a
                -- job.

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

jobComment(1    -- Octets:  An arbitrary human-readable coded
8),             -- 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.


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

jobPriority(    -- Integer32(0..100):  The priority for
19)             -- scheduling the job. It is used by servers
                -- and devices that employ a priority-based
                -- scheduling algorithm.
                --
                -- A higher value specifies a higher priority.
                -- The value 1 is defined to indicate the
                -- lowest possible priority (a job which a
                -- priority-based scheduling algorithm shall
                -- pass over in favor of higher priority jobs).
                -- The value 100 is defined to indicate the
                -- highest possible priority. Priority is
                -- expected to be evenly or 'normally'
                -- distributed across this range. The mapping
                -- of vendor-defined priority over this range
                -- is implementation-specific.
                --
                -- A value of 0 shall be returned by
                -- implementations that do not have a priority-
                -- based queuing algorithm.

jobProcessAf    -- Integer:  The calendar date and time of day
terDateAndTi    -- after which the job shall become a candidate
me(20)          -- 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 and the job's jobState
                -- attribute to held and add the
                -- jobProcessAfterSpecified bit value to the
                -- job's jobStateReasons attribute 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 jobStateReasons attribute and, if no
                -- other reasons remain, shall change the job's
                -- jmJobState and the job's jobState attribute
                -- to pending so that the job becomes a
                -- candidate for being scheduled on devices(s).
                --
                -- The server 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.

outputBinInd    -- Integer:  The output subunit index in the
ex(21),         -- Printer MIB of the output bin to which all
                -- or part of the job is placed in.
                --
                -- A row with this attribute item may appear
                -- more than once in the jmAttributeTable for a
                -- job, but the jmAttributeValueAsInteger shall
                -- be different for each such row.

outputBinNam    -- Octets:  The name of the output bin to which
e(22),          -- all or part of the job is placed in.
                --
                -- A row with this attribute item may appear
                -- more than once in the jmAttributeTable for a
                -- job, but the jmAttributeValueAsOctets shall
                -- be different for each such row.

sides(23),      -- Integer:  The number of sides that any
                -- document in this job will require or did
                -- use.

documentForm    -- Integer:  The interpreter language family
atIndex(24),    -- index in the Printer MIB of the
                -- prtInterpreterLangFamily object, that this
                -- job requires and uses.  A document or a job
                -- may use more than one PDL.
                --
                -- A row with this attribute item may appear
                -- more than once in the jmAttributeTable for a
                -- job, but the jmAttributeValueAsInteger shall
                -- be different for each such row.  As with all
                -- intensive attribute items where multiple
                -- rows are allowed, there shall be only one
                -- distinct row for each distinct PDL; there
                -- shall be no duplicates.
                --
                -- NOTE - This attribute type is intended to be
                -- used with an agent that implements the
                -- Printer MIB and shall not be used if the
                -- agent does not implement the Printer MIB.
                -- Such as agent shall use the
```

```
                -- documentFormatEnum attribute instead.

documentForm    -- Integer:  The interpreter language family
atEnum(25),     -- corresponding to the Printer MIB
                -- prtInterpreterLangFamily object, that this
                -- job requires and uses.  A document or a job
                -- may use more than one PDL.
                --
                -- A row with this attribute item may appear
                -- more than once in the jmAttributeTable for a
                -- job, but the jmAttributeValueAsInteger shall
                -- be different for each such row.  As with all
                -- intensive attribute items where multiple
                -- rows are allowed, there shall be only one
                -- distinct row for each distinct PDL; there
                -- shall be no duplicates.
                --
                -- This enum is a type 2 enum.
                --
                -- NOTE: The PrtInterpreterLangFamilyTC textual
                -- convention is defined in the draft Printer
                -- MIB, but is not in RFC 1759.


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

    jobCopiesReq  -- Integer:  The number of copies of the entire
    uested(26),   -- job that are to be produce
                  --
                  -- A value of -2 means unknown.

    jobCopiesCom  -- Integer:  The number of copies of the entire
    pleted(27),   -- job that the entire job has completed so
                  -- far.
                  --
                  -- A value of (-2) means unknown.

    documentCopi  -- Integer:  The total count of the number of
    esRequested(  -- document copies requested.  If there are
    28),          -- documents A, B, and C, and document B is
                  -- specified to produce 4 copies, the number of
                  -- document copies requested is 6 for the job.

    documentCopi  -- Integer:  The total count of the number of
    esCompleted(  -- document copies completed so far for the job
    29),          -- 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.

jobKOctetsRe    -- Integer:  The total number of K (1024)
quested(30),    -- octets being requested to be processed in
                -- the job, including document and job copies.
                -- The agent shall round the actual number of
                -- octets up to the next highest K.  Thus 0
                -- octets shall be represented as 0, 1-1024
                -- octets shall be represented as 1, 1025-2048
                -- shall be represented as 2, etc.
                --
                -- The server/device may update the value of
                -- this attribute after each document has been
                -- transferred to the server/device or the
                -- server/device may provide this value after
                -- all documents have been transferred to the
                -- server/device, depending on implementation.
                -- In other words, while the job is in the held
                -- state with the jobStateReasons attribute
                -- containing a documentsNeeded or
                -- preProcessing value, the value of the
                -- jobKOctetsRequested attribute depends on
                -- implementation and may not correctly reflect
                -- the size of the job.
                --
                -- In computing this value, the server/device
                -- shall include the multiplicative factors
                -- contributed by (1) the number of document
                -- copies, and (2) the number of job copies,
                -- independent of whether the device can
                -- process multiple copies of the job or
                -- document without making multiple passes over
                -- the job or document data and independent of
                -- whether the output is collated or not.  Thus
                -- the server/device computation is independent
                -- of the implementation and shall be:
                --
                --     (1) Document contribution:  Multiply the
                --     size of each document in octets by the
                --     number of document copies of that
                --     document.
                --
                --     (2) Add each document contribution
                --     together.
                --
                --     (3) Job copy contribution:  Multiply the
                --     job size by the number of job copies.
                --
                --     (4) Round up the result to the next
                --     higher K (1024 multiple).
                --
                -- The total K octets to be processed can be
```

```
                -- used in the denominator with the
                -- jobKOctetsCompleted attribute in the
                -- numerator in order to produce a
                -- 'thermometer' that indicates the progress of
                -- the job.
                --
                -- The value (-2) means unknown.
                --
jobKOctetsCo    -- Integer:  The number of K (1024) octets
mpleted(31),    -- currently processed by the server or device,
                -- including document and job copies.  For
                -- printing, the completed count only includes
                -- processing (interpreting) if the
                -- implementation distinguishes between the
                -- processing and printing states; otherwise,
                -- the completed count includes both processing
                -- (interpreting) and marking combined
                -- together.  For scanning, the completed count
                -- only includes scanning, if the
                -- implementation distinguishes between the
                -- processing and (to be registered) scanning
                -- states; otherwise the completed count
                -- includes both scanning and processing
                -- (formatting).
                --
                -- The agent shall round the actual number of
                -- octets completed up to the next higher K.
                -- Thus 0 octets is represented as 0, 1-1023,
                -- is represented as 1, 1024-2047 is 2, etc.
                -- When the job completes, the values of the
                -- jobKOctetsRequested and the
                -- jobKOctetsCompleted attributes shall be
                -- equal.
                --
                -- For multiple copies generated from a single
                -- data stream, the value shall be incremented
                -- as if each copy was printed from a new data
                -- stream without resetting the count between
                -- copies.  See the pagesCompletedCurrentCopy
                -- attribute that is reset on each document
                -- copy.
                --
                -- The total K octets completed can be used in
                -- the numerator with the jobKOctetsRequested
                -- attribute in the denominator in order to
                -- produce a "thermometer" that indicates the
                -- progress of the job.
                --
                -- The value of this attribute shall be 0 if
                -- processing has not started for this job.

    -- ************************************************************
```

```
-- Impression attributes
--
-- For a print job, an impression is the marking of the
-- entire side of a sheet.  Two-sided processing involves two
-- impressions per sheet.  Two-up is the placement of two
-- logical pages on one side of a sheet and so is still a
-- single impression.
-- ************************************************************

impressionsS  -- Integer:  The number of impressions spooled
pooled(32),   -- to the server or device for the job so far.

impressionsS  -- Integer:  The number of impressions sent to
entToDevice(  -- the device for the job so far.
33),
impressionsI  -- Integer:  The number of impressions
nterpreted(3  -- interpreted for the job so far.
4),
impressionsR  -- Integer:  The number of impressions
equested(35)  -- requested by this job to produce.
,
impressionsC  -- Integer:  The total number of impressions
ompleted(36)  -- 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.
              --
              -- The value of this attribute shall be 0 if
              -- processing has not started for this job.

impressionsC  -- Integer:  The number of impressions
ompletedCurr  -- completed by the device for the current copy
entCopy(37),  -- 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.
              --
              -- The value of this attribute shall be 0 if
              -- processing has not started for this job.


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

```
-- ********************************************************

pagesRequested(38),     -- Integer:  The number of logical pages
                        -- requested by the job to be processed.

pagesCompleted(39),     -- Integer:  The total number of logical pages
                        -- completed for this job so far.

pagesCompletedCurrentCopy(40),  -- 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(41),    -- Integer:  The total number of medium sheets
                        -- requested to be processed for this job.

sheetsCompleted(42),    -- Integer:  The total number of medium sheets
                        -- that have completed marking and stacking for
                        -- the entire job so far whether those sheets
                        -- have been processed on one side or on both.
                        -- The value of this attribute shall be 0 if
                        -- processing has not started for this job.

sheetsCompletedCurrentCopy(43),  -- 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 each document in the job and for each
                        -- document copy.

mediumRequested(44),    -- Octets:  The name of the medium that is
                        -- required by the job.
                        --
                        -- A row with this attribute item may appear
                        -- more than once in the jmAttributeTable for a
                        -- job, but the jmAttributeValueAsOctets shall
                        -- be different for each such row.

mediumConsumed(45),     -- Octets: 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.
                --
                -- A row with this attribute item may appear
                -- more than once in the jmAttributeTable for a
                -- job, but the jmAttributeValueAsOctets shall
                -- contain a different name for each such row.
                --
                -- The value of this attribute shall be 0 if
                -- processing has not started for this job.

colorantRequ    -- Integer:  The index (prtMarkerColorantIndex)
estedIndex(4    -- in the Printer MIB of the colorant
6),             -- requested.
                --
                -- A row with this attribute item may appear
                -- more than once in the jmAttributeTable for a
                -- job, but the jmAttributeValueAsOctets shall
                -- be different for each such row.

colorantRequ    -- Octets:  The name of the colorant requested.
estedName(47    --
),              -- A row with this attribute item may appear
                -- more than once in the jmAttributeTable for a
                -- job, but the jmAttributeValueAsOctets shall
                -- be different for each such row.

colorantCons    -- Integer:  The index (prtMarkerColorantIndex)
umedIndex(48    -- in the Printer MIB of the colorant consumed.
),              --
                -- A row with this attribute item may appear
                -- more than once in the jmAttributeTable for a
                -- job, but the jmAttributeValueAsOctets shall
                -- be different for each such row.

colorantCons    -- Octets:  The name of the colorant consumed.
umedName(49)    --
,               -- A row with this attribute item may appear
                -- more than once in the jmAttributeTable for a
                -- job, but the jmAttributeValueAsOctets shall
                -- be different for each such row.


-- ********************************************************
-- Time attributes
--
-- Two forms of time are provided.  Each form is represented
-- in a separate attribute.  Implementations may choose the
-- more appropriate form.  An implementation need not provide
-- both forms and is recommended not to provide both forms
-- for a particular attribute.  However, some attributes may
-- be in one form and others may be in the other form,
```

```
-- depending on the source of the time.  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.

-- JmTimeTC is the time of day measured in the number of
-- seconds as an offset from the integer value of sysUpTime
-- (which is measured in hundredths of a second).
-- See page 22.
-- ********************************************************

jobSubmissio  -- Octets:  The date and time that the job was
nDateAndTime  -- submitted.  The value shall be specified
(50),         -- using the DateAndTime textual convention
              -- from SMIv2-TC.

jobSubmissio  -- Integer:  The time that the job was
nTime(51),    -- submitted.  The value shall be specified
              -- using the JmTimeTC textual convention (see
              -- page 22).

jobStartedBe  -- Integer:  The time that the job started
ingHeldTime(  -- being held, i.e., the time that the job
52),          -- entered the held state most recently.  The
              -- value shall be specified using the JmTimeTC
              -- textual convention (see page 22).  If the
              -- job has never entered the held state, then
              -- the value shall be 0.

jobStartedPr  -- Octets:  The date and time that the job
ocessingDate  -- started processing.  The value shall be
AndTime(53),  -- specified using the DateAndTime textual
              -- convention from SMIv2-TC.

jobStartedPr  -- Integer:  The time that the job started
ocessingTime  -- processing.  The value shall be specified
(54),         -- using the JmTimeTC textual convention (see
              -- page 22).

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

jobCompleted  -- Integer:  The time that the job completed
Time(56),     -- processing and the medium is completely
```

```
                    -- stacked in the output bin, i.e., when the
                    -- job entered the completed state.  The value
                    -- shall be specified using the JmTimeTC
                    -- textual convention (see page 22).

      processingCP  -- Integer:  The amount of CPU time that the
      UTime(57)     -- job has been processing in seconds.  If the
                    -- job needs attention, that elapsed time shall
                    -- not be included.  In other words, the
                    -- processingCPUTime should be relatively
                    -- repeatable.
                    --
                    -- The value of this attribute shall be 0 if
                    -- processing has not started for this job.

    }



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 values: 8+32+4=44, respectively.

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

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

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

    -- This is a type 2 enumeration.  See Section 7.1 on page 18.
    SYNTAX      INTEGER {
        other(1),       -- The job contains some document production
                        -- instructions that are not one of the
```

```
                        -- identified types.

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

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

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

            faxIn(16),      -- The job contains some document production
                            -- instructions that specify receive fax

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

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

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

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

947         }
948
949
950
951
952     JmJobStateReasonsTC ::= TEXTUAL-CONVENTION
953         STATUS      current
954         DESCRIPTION
955             "This textual-convention is used in the jobStateReasons
956             attribute to provides additional information regarding the
957             jmJobState object and the jobState attribute.  The
958             jobStateReasons attribute identifies the reason or reasons that
959             the job is in the held, pending, processing, printing,
960             needsAttention, canceled, or completed state.  The server shall
961             indicate the particular reason(s) by setting the value of the
962             jobStateReasons attribute.  While the job states cannot be added
963             to without impacting deployed clients, it is the intent that
964             additional JmJobStateReasonsTC enums can be defined without
965             impacting deployed clients.  In other words, the
966             JmJobStateReasonsTC is intended to be extensible.
```

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

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

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

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

This object is a type 2 bit string. See Section 7 entitled 'IANA Considerations' on page 18.

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

```
-- This is a type 2 bit string.  See section 7.2 on page 19.
SYNTAX      INTEGER {
--   really OCTET STRING(SIZE(0..63))
    documentsNeeded(1),     -- The job is in the held state because
                            -- the server or printer is waiting for
                            -- the job's files to start and/or finish
                            -- being transferred before the job can
                            -- be scheduled to be printed.

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

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

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

```
                        -- which the job is a candidate.

successfulCompletion    -- The job is in the completed state
(5),                    -- having completed successfully.
                        --
completedWithWarning    -- The job is in the canceled or
s(6),                   -- completed states having completed with
                        -- warnings.

completedWithErrors(    -- The job is in the canceled or
7),                     -- completed states having completed with
                        -- errors (and possibly warnings too).
                        --
canceledByUser(8),      -- The job is in the canceled,  state
                        -- having been canceled by the user.
                        --
canceledByOperator(9    -- The job is in the canceled state
),                      -- having been canceled by the operator.

abortedBySystem(10),    -- The job is in the canceled, state
                        -- having been aborted by the system.

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

logfileTransferring(    -- The job is in the canceled or
12),                    -- completed states and the job's logfile
                        -- is being transferred.

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

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

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

```
                                -- job-retention-period would be set to 0
                                -- and the job is deleted].

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

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

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

```
                        -- failure value.

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

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

jobStartWaitTimeOut(    -- The server/device has stopped the job
21),                    -- at the beginning of processing to
                        -- await human action, such as installing
                        -- a special cartridge or special non-
                        -- standard media, but the job was not
                        -- resumed within the site-settable time-
                        -- out value and the server/device has
                        -- transitioned the job to the held
                        -- state.  Normally, the job is resumed
                        -- by means outside the job submission
                        -- protocol, such as some local function
                        -- on the device.

jobEndWaitTimeOut(22    -- The server/device has stopped the job
),                      -- at the end of processing/printing 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 canceled
                        -- state.

jobPasswordWaitTimeO    -- The server/device has stopped the job
ut(23),                 -- at the beginning of processing to
                        -- await input of the job's password, but
```

```
                                -- the human intervention did not occur
                                -- within the site-settable time-out
                                -- value and the server/device has
                                -- transitioned the job to the held
                                -- state.  Normally, the password is
                                -- input and the job is resumed by means
                                -- outside the job submission protocol,
                                -- such as some local function on the
                                -- device.

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

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

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

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

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

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

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

```
validating(31),          -- The server/device is validating the
                         -- job after accepting the job.  The job
                         -- state may be held, pending, or
                         -- processing.

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

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

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

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

noSpaceOnServer(36),     -- The job is held because there is no
                         -- room on the server to store all of the
                         -- job.  For example, there is no room
                         -- for the document data or a scan-to-
                         -- file job.

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

exceededAccountLimit     -- The account for which this job is
(38),                    -- drawn has exceeded its limit.  This
                         -- condition should be detected before
                         -- the job is scheduled so that the user
                         -- does not wait until his/her job is
                         -- scheduled only to find that the
                         -- account is overdrawn.  This condition
                         -- may also occur while the job is
                         -- processing either as processing begins
```

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

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


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

                                -- The job was canceled because the
                                -- server or device was shutdown before
                                -- completing the job.  The job shall be
                                -- placed in the pending state [if the
        canceledByShutdown(4    -- job was not started, else the job
        0),                     -- shall be placed in the terminating
                                -- state].

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

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

```
                                -- job back to a physical device that has
                                -- rejected the job for this job-state-
                                -- reasons value. Otherwise, [the job]
                                -- shall be placed in the completed state
                                -- with the retained value set in the
                                -- jobStateReasons attribute.

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

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


-- The following additional job state reasons have been added to
-- specify sub-states of the held state that are in ISO DPA::

        jobPreProcessing(45)    -- The job has been created on the server
        ,                       -- or device but the submitting client is
                                -- in the process of adding additional
                                -- job components and no documents have
                                -- started processing.  The job maybe in
                                -- the process of being checked by the
                                -- server/device for attributes, defaults
                                -- being applied, a device being
                                -- selected, etc.

        jobPaused(46),          -- The job has been indefinitely
                                -- suspended by a client issuing an
                                -- operation to suspend the job so that
                                -- other jobs may proceed using the same
                                -- devices.  The client may issue an
                                -- operation to resume the paused job at
                                -- any time, in which case the server or
                                -- printer places the job in the held or
                                -- pending states and the job is
                                -- eventually resumed at the point where
                                -- the job was paused.

        jobInterrupted(47),     -- The job has been interrupted while
                                -- processing by a client issuing an
                                -- operation that specifies another job
                                -- to be run instead of the current job.
```

```
                                -- The server or printer will
                                -- automatically resume the interrupted
                                -- job when the interrupting job
                                -- completes.

            jobRetained(48)     -- The job is being retained by the
                                -- server or printer after processing and
                                -- all of the media have been
                                -- successfully stacked in the output
                                -- bin(s).
                                --
                                -- The job (1) has completed successfully
                                -- or with warnings or errors, (2) has
                                -- been aborted while printing by the
                                -- server/device, or (3) has been
                                -- canceled by the submitting user or
                                -- operator before or during processing.
                                -- The job's jobStateReasons attribute
                                -- shall contain the reasons that the job
                                -- has entered the retained sub-state of
                                -- the completed state.
                                --
                                -- While in the retained state, all of
                                -- the job's document data (and submitted
                                -- resources, such as fonts, logos, and
                                -- forms, if any) are retained by the
                                -- server or device; thus a client could
                                -- issue an operation to resubmit the job
                                -- (or a copy of the job) while the job
                                -- is in the retained state.
                                --
                                -- The retained state is conditionally
                                -- mandatory.  Implementations that do
                                -- not retain jobs after they are
                                -- finished processing such that the
                                -- client could request that the job be
                                -- repeated (or resubmitted), need not
                                -- implement the retained state.

    }


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

Table 1 - Legal Job States for each Job State Reason

--	Descriptive Name	Allowed job states
--	**documents-needed(1)**	**held**
--	**job-hold-set(2)**	**held**

--	Descriptive Name	Allowed job states
--	job-process-after-specified(3)	held
--	required-resources-not-ready(4)	held
--	successful-completion(5)	completed
--	completed-with-warnings(6)	completed
--	completed-with-errors(7)	completed
--	canceled-by-user(8)	canceled
--	canceled-by-operator(9)	canceled
--	aborted-by-system(10)	canceled
--	logfile-pending(11)	canceled
--	logfile-transferring(12)	canceled
--	cascaded(13)	canceled
--	deleted-by-administrator(14)	canceled
--	discard-time-arrived(15)	canceled
--	postprint-failed(16)	canceled, completed
--	submission-interrupted(17)	canceled
--	max-job-fault-count-exceeded(18)	canceled
--	devices-need-attention-time-out(19)	held, canceled
--	needs-key-operator-time-out(20)	held, canceled
--	job-start-wait-time-out(21)	canceled
--	job-end-wait-time-out(22)	canceled
--	job-password-wait-time-out(23)	held, pending
--	device-timed-out(24)	held, canceled
--	connecting-to-device-time-out(25)	held, canceled
--	transferring(26)	processing
--	queued-in-device(27)	processing
--	job-cleanup(28)	processing
--	processing-to-stop-point(29)	processing
--	job-password-wait(30)	held, processing
--	validating(31)	held, pending, processing
--	queue-held(32)	held
--	job-proof-wait(33)	held
--	held-for-diagnostics(34)	held
--	service-off-line(35)	held
--	no-space-on-server(36)	held
--	pin-required(37)	held, canceled
--	exceeded-account-limit(38)	held, canceled
--	held-for-retry(39)	held
--	canceledByShutdown(40)	canceled
--	deviceUnavailable(41)	pending
--	wrongDevice(42)	canceled
--	badJob(43)	canceled
--	jobInterruptedByDeviceFailure(44)	held
--	jobPreProcessing(45)	held
--	jobPaused(46)	held
--	jobInterrupted(47)	held
--	jobRetained(48)	completed

```
-- 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 16.

jmGeneral  OBJECT IDENTIFIER ::= { jobmonmib 5 }

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 10 for the definition of a job set.

        The jmGeneralTable which is indexed by:

        1.  jmJobSetIndex -  a running index of Job Set instances
            supported by this device or server.  A job set is used in
            the MIB to represent the separation of jobs into disjoint
            sets for scheduling purposes in a server, typically into
            separate job queues.  See Terminology and Job Model on page
            10 for the definition of a job set."
    ::= { jmGeneral 1 }

jmGeneralEntry  OBJECT-TYPE
    SYNTAX      JmGeneralEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "Information about a job set (queue).  See Terminology and Job
        Model on page 10 for the definition of a job set.

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

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

jmGeneralJobSetName OBJECT-TYPE
    SYNTAX      OCTET STRING(SIZE(0..63))
    MAX-ACCESS  read-only
```

```
    STATUS      current
    DESCRIPTION
        "The human readable administratively assigned name of this job
        set.  Typically, this name will be the name of the job queue.
        If a server or printer has only a single job set, this object
        can be the administratively assigned name of the server or
        printer itself.  This name does not need to be unique, though
        each job set in a single Job Monitoring MIB should have distinct
        names.

        The purpose of this object is to help the user of the job
        monitoring application distinguish between several job sets in
        implementations that support more than one job set."
    ::= { jmGeneralEntry 1 }

jmGeneralJobPersistence OBJECT-TYPE
    SYNTAX      Integer32(0..2147483647)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The minimum time in seconds that an entry will remain in the
        jmJobIDTable and jmJobStateTable after processing/printing has
        completed as specified by the system administrator or the
        implementation for this instance of the Job Set."
    ::= { jmGeneralEntry 2 }

jmGeneralAttributePersistence OBJECT-TYPE
    SYNTAX      Integer32(0..2147483647)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The minimum time in seconds that an entry will remain in the
        jmAttributeTable after processing/printing has completed, i.e.,
        the time in seconds starting when the job enters the completed
        or canceled 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 for this
        instance of the Job Set, depending on implementation.  This
        value shall be equal to or less than the value of
        jmGeneralJobPersistence.  Attributes that are shared between the
        jmJobIDTable/jmJobStateTable and the jmAttributeTable shall be
        governed by the larger value in all tables."
    ::= { jmGeneralEntry 3 }

jmGeneralNumberOfActiveJobs OBJECT-TYPE
    SYNTAX      Integer32(0..2147483647)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The current number of active jobs in the jmJobIDTable,
        jmJobStateTable, and jmAttributeTable, i.e., the total number of
        jobs that have neither completed nor have been canceled.  See
```

```
        JmJobStateTC on page 22 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 4 }

jmGeneralOldestActiveJobIndex  OBJECT-TYPE
    SYNTAX      Integer32 (0..2147483647)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The jmJobIndex of the oldest active job, i.e., the job in the
        jmJobStateTable and jmAttributeTable that has been there the
        longest and has neither completed nor been canceled.

        If there are no active jobs, the value shall be 0.

        NOTE - For implementations that process jobs in order of
        submission, this object indicates the 'separating line' between
        completed jobs and jobs that are still active.  However, an
        application shall still have to skip over canceled jobs when
        searching for active jobs.

        NOTE - Applications that wish to skip over completed or canceled
        jobs may use this value to start with the oldest active job and
        continue until they reach the index value equal to
        jmGeneralNewestActiveJobIndex, skipping over any completed or
        canceled jobs that might intervene.  Since jobs may arrive while
        such an application is performing GetNext operations, the
        application should always get the value of
        jmGeneralNewestActiveJobIndex in each GetNext operation to see
        if this job is still the newest.  If an application gets the no
        more rows ??? return, the job index may have wrapped such that
        the jmGeneralNewestActiveJobIndex is smaller than
        jmGeneralOldestActiveJobIndex.  In this case, the application
        shall start over at 1 and continue the GetNext operations to
        find the rest of the active jobs."
    ::= { jmGeneralEntry 5 }

jmGeneralNewestActiveJobIndex  OBJECT-TYPE
    SYNTAX      Integer32 (0..2147483647)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The jmJobIndex of the newest active job, i.e., the job in the
        jmJobStateTable and jmAttributeTable that has been added most
        recently and has neither completed nor been canceled.

        If there are no active jobs, the value shall be 0."
    ::= { jmGeneralEntry 6 }
```

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

-- The jmJobIDGroup consists entirely of the jmJobIDTable.
--
-- The two key indexes that are used in other tables to index jobs:
-- jmJobSetIndex and jmJobIndex are materialized in this group.
--
-- Implementation of every object in this group is mandatory.
-- See Section 4 entitled 'Conformance Considerations' on page 16.

jmJobID  OBJECT IDENTIFIER ::= { jobmonmib 6 }

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

        See Terminology and Job Model on page 10 for the definition of a
        job set.

        The jmJobIDTable is indexed by:

        1.  jmJobSubmissionIDIndex -  a 32-octet job identifier
            generated when the job was submitted, either by the client
            or the server/printer.
    ::= { jmJobID 1 }

jmJobIDEntry  OBJECT-TYPE
    SYNTAX      JmJobIDEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The map from (1) the jmJobSubmissionIDIndex 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."
    INDEX  { jmJobSubmissionIDIndex }
    ::= { jmJobIDTable 1 }

JmJobIDEntry ::= SEQUENCE {
    jmJobSubmissionIDIndex            OCTET STRING(SIZE(0..63)),
```

```
        jmJobSetIndex                   Integer32(1..2147483647),
        jmJobIndex                      Integer32(1..2147483647),
}

jmJobSubmissionIDIndex OBJECT-TYPE
    SYNTAX      OCTET STRING(SIZE(0..63))
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A quasi-unique 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 jmJobSubmissionIDIndex.  Each
        format shall be registered using the procedures of a type 2
        enum.  See section entitled: 'IANA Registration of enums' on
        page 18.

        The value of jmJobSubmissionIDIndex should be one of the
        registered format types.  The first two octets of the string
        shall indicate which registered format is being used.  The agent
        shall assign a string of registered format (00) for any job
        without a value.  The format values registered so far are:

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

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

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

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

          04       to be registered according to procedures of a type 2
                   enum.

        NOTE - the job submission id 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
```

```
1260            or URL, a sequential number should suffice for each client (and
1261            may be easier for each client to manage).  Therefore, the length
1262            of the job submission id has been selected to reduce the
1263            probability of collision to a very low number, but is not
1264            intended to be an absolute guarantee of uniqueness.  None-the-
1265            less, collisions could occur, but without bad consequences,
1266            since this MIB is intended to be used only for monitoring jobs,
1267            not for controlling and managing them."
1268        ::= { jmJobIDEntry 1 }
1269
1270    jmJobSetIndex OBJECT-TYPE
1271        SYNTAX      Integer32(1..2147483647)
1272        MAX-ACCESS  read-only
1273        STATUS      current
1274        DESCRIPTION
1275            "The job set index of the job set in which the job was placed
1276            when that server or device accepted the job.  This value in
1277            combination with the jmJobIndex value permits the management
1278            application to access the other tables to obtain the job-
1279            specific objects.  This value shall be the same for a job in the
1280            jmJobIDTable as the corresponding jmJobSetIndex value in the
1281            jmJobStateTable and jmAttributeTable for this job.
1282
1283            NOTE - an implementation that has only one job set, such as a
1284            printer with a single queue, shall hard code this object with
1285            the value 1.  See Terminology and Job Model on page 10 for the
1286            definition of a job set."
1287        ::= { jmJobIDEntry 2 }
1288
1289    jmJobIndex OBJECT-TYPE
1290        SYNTAX      Integer32(1..2147483647)
1291        MAX-ACCESS  read-only
1292        STATUS      current
1293        DESCRIPTION
1294            "The sequential, monatonically increasing identifier for the job
1295            generated by the server or device when that server or device
1296            accepted the job.  This value permits the management application
1297            to access the other tables to obtain the job-specific objects.
1298            This value shall be the same for a job in the jmJobIDTable as
1299            the corresponding jmJobIndex value in the jmJobStateTable and
1300            jmAttributeTable for this job.
1301
1302            The value 0 shall not be generated.  Agents instrumenting
1303            systems that contain jobs with a job identifier of 0 shall map
1304            the value 0 to a value that is one higher than the highest job
1305            identifier value that any job can have on that system."
1306        ::= { jmJobIDEntry 3 }
1307
1308
1309
1310
1311    -- The Job State Group (Mandatory)
1312
```

```
-- The jmJobStateGroup consists entirely of the jmJobStateTable.
--
-- Implementation of every object in this group is mandatory.
-- See Section 4 entitled 'Conformance Considerations' on page 16.

jmJobState  OBJECT IDENTIFIER ::= { jobmonmib 7 }

jmJobStateTable  OBJECT-TYPE
    SYNTAX      SEQUENCE OF JmJobStateEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The jmJobStateTable consists of basic job state and status
        information for each job in a job set that (1) monitoring
        applications need to be able to access in a single SNMP Get
        operation, (2) that have a single value per job, and (3) that
        shall always be implemented.  See Terminology and Job Model on
        page 10 for the definition of a job set.

        NOTE - Every accessible object in this table shall have the same
        value as one of the attributes in the jmAttributeTable.
        Implementations may either keep a separate copy or may share
        each value that is common between the jmJobStateTable and the
        jmAttributeTable.  The persistence of the two tables may be
        different depending on implementation and/or system
        administrator policy as specified by the jmGeneralJobPersistence
        and jmGeneralAttributePersistence objects defined on page 55.
        Thus an accounting application need only copy the entire
        jmAttributeTable or selected job rows and will obtain all of the
        information about the job and its state.

        The jmJobStateTable is indexed by:

        1.  jmJobSetIndex -  a running index of Job Set instances
            supported by this device or server.  A job set is used in
            the MIB to represent the separation of jobs into disjoint
            sets for scheduling purposes in a server, typically into
            separate job queues.  See Terminology and Job Model on page
            10 for the definition of a job set.

        2.  jmJobIndex -  the job identifier that was generated by the
            server or device that accepted the job."
    ::= { jmJobState 1 }

jmJobStateEntry  OBJECT-TYPE
    SYNTAX      JmJobStateEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "Basic per-job state and status information.
```

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

    JmJobStateEntry ::= SEQUENCE {
        jmJobState                              JmJobStateTC,
        jmJobStateKOctetsCompleted              Integer32(0..2147483647),
        jmJobStateImpressionsCompleted          Integer32(0..2147483647),
        jmJobStateAssociatedValue               Integer32(0..2147483647)
    }



    jmJobState OBJECT-TYPE
        SYNTAX      JmJobStateTC   -- See page 21
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "The current state of the job (pending, processing, held, etc.).

            The value of this object shall always be the same as that of the
            jobState attribute, so that this information appears in both the
            jmJobStateTable and the jmAttributeTable simultaneously.  See
            the JmJobStateTC textual-convention on page 21 and the jobState
            attribute on page 27 in the jmAttributeTable for the full
            specification of this object/attribute.

            This object is a type 2 enum."
        ::= { jmJobStateEntry 1 }


    jmJobStateKOctetsCompleted OBJECT-TYPE
        SYNTAX      Integer32(0..2147483647)
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "The current number of octets completed processing by the server
            or device measured in units of K (1024) octets.

            The value of this object shall always be the same as that of the
            jobKOctetsCompleted attribute, so that this information appears
            in both the jmJobStateTable and the jmAttributeTable
            simultaneously.  See the jobKOctetsCompleted attribute on page
            37 in the jmAttributeTable for the full specification of this
            object/attribute."
        ::= { jmJobStateEntry 2 }

    jmJobStateImpressionsCompleted OBJECT-TYPE
        SYNTAX      Integer32(0..2147483647)
        MAX-ACCESS  read-only
        STATUS      current
```

```
1413        DESCRIPTION
1414            "The current number of impressions completed being marked and
1415            stacked by the device for this job so far.
1416
1417            The value of this object shall always be the same as that of the
1418            impressionsCompleted attribute, so that this information appears
1419            in both the jmJobStateTable and the jmAttributeTable
1420            simultaneously.  See the impressionsCompleted attribute on page
1421            38 in the jmAttributeTable for the full specification of this
1422            object/attribute."
1423        ::= { jmJobStateEntry 3 }
1424
1425    jmJobStateAssociatedValue OBJECT-TYPE
1426        SYNTAX      Integer32(0..2147483647)
1427        MAX-ACCESS  read-only
1428        STATUS      current
1429        DESCRIPTION
1430            "The value of the most relevant attribute associated with the
1431            job's current state.
1432
1433            The value of this object shall always be the same as that of the
1434            jobStateAssociatedValue attribute, so that this information
1435            appears in both the jmJobStateTable and the jmAttributeTable
1436            simultaneously.  See the jobStateAssociatedValue attribute on
1437            page 27 in the jmAttributeTable for the full specification of
1438            this object/attribute."
1439        ::= { jmJobStateEntry 4 }
1440
1441
1442
1443    -- The Attribute Group (Mandatory)
1444
1445    -- The jmAttributeGroup consists entirely of the jmAttributeTable.
1446    --
1447    -- Implementation of every object in this group is mandatory.
1448    -- See Section 4 entitled 'Conformance Considerations' on page 16.
1449    --
1450    -- Some attributes are mandatory for conformance, and the rest are
1451    -- optional.  The mandatory attributes are the ones required to have
1452    -- copies in the jmJobStateTable.  The mandatory attributes are:
1453    --
1454    --    jobState
1455    --    numberOfInterveningJobs
1456    --    deviceAlertCode
1457    --    jobKOctetsRequested
1458    --    jobKOctetsCompleted
1459    --    impressionsRequested
1460    --    impressionsCompleted
1461    --    outputBinName
1462
1463
1464    jmAttribute  OBJECT IDENTIFIER ::= { jobmonmib 8 }
1465
```

```
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.  Some
        attributes represent information about the job and document(s),
        such as file-names, document-names, submission-time, completion-
        time, size, etc.  Other attributes represent requested and/or
        consumed resources for each job.

        The jmAttributeTable is a per-job table with an extra index for
        each type of attribute (jmAttributeTypeIndex) that a job can
        have and an additional index (jmAttributeInstanceIndex) for
        those attributes that can have multiple instances per job.  The
        jmAttributeTypeIndex object shall contain an enum type that
        indicates the type of attribute (see JmAttributeTypeTC on page
        25).  The value of the attribute shall be represented in either
        the jmAttributeValueAsInteger or jmAttributeValueAsOctets
        objects, or both, as specified in the JmAttributeTypeTC on page
        25.

        The agent shall create rows in the jmAttributeTable as the
        server or printer 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 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 jmAttributeTable is indexed by (from most significant to
        least significant):

        1) jmJobSetIndex -  a running index of Job Set instances
           supported by this device or server.  A job set is used in the
           MIB to represent the separation of jobs into disjoint sets
           for scheduling purposes in a server, typically into separate
           job queues.  See Terminology and Job Model on page 10 for the
           definition of a job set.

        2) jmJobIndex -  the job identifier that was generated by the
           server or device that accepted the job.

        3) jmAttributeTypeIndex - the enum that indicates the type of
           the attribute.  See JmAttributeTypeTC on page 25 for the
           specification of each attribute.
```

```
            4) jmAttributeInstanceIndex -  a running index of attributes of
               the same type for each job.  For those attributes with only a
               single instance per job, this index value shall be 1.  For
               those attributes that are a single value per document, the
               index value shall be the document number, starting with 1 for
               the first document in the job.  Jobs with only a single
               document shall use the index value of 1.  For those
               attributes that can have multiple values per job and per
               document, such as documentFormatIndex or documentFormatEnum,
               the index shall be a running index for the job as a whole,
               starting at 1."
        ::= { jmAttribute 1 }

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

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

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

    jmAttributeTypeIndex OBJECT-TYPE
        SYNTAX      JmAttributeTypeTC   -- See page 25
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
            "The type of attribute.

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

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

            Examples of resources required and consumed include:
            jobKOctetsRequested, jobKOctetsCompleted, pagesRequested,
            pagesCompleted, mediumRequested, and mediumConsumed,
```

```
1567            respectively.  See the JmAttributeTypeTC textual convention on
1568            page 25.
1569
1570            In the definitions of the enums in the JmAttributeTypeTC textual
1571            convention, each description indicates whether the value of the
1572            attribute shall be represented using the
1573            jmAttributeValueAsInteger or the jmAttributeValueAsOctets
1574            objects by the initial tag: 'Integer:' or 'Octets:',
1575            respectively.  A very few attributes use both objects
1576            (mediumConsumed)and so have both tags.
1577
1578            If the jmAttributeValueAsInteger object is not used (no
1579            'Integer:' tag), the agent shall return the value (-1)
1580            indicating other.  If the jmAttributeValueAsOctets object is not
1581            used (no 'Octets:' tag), the agent shall return a zero-length
1582            octet string.
1583
1584            This value is a type 2 enum."
1585        ::= { jmAttributeEntry 1 }
1586
1587    jmAttributeInstanceIndex OBJECT-TYPE
1588        SYNTAX      Integer32(1..32767)
1589        MAX-ACCESS  not-accessible
1590        STATUS      current
1591        DESCRIPTION
1592            "A running 16-bit index of the attributes of the same type for
1593            each job.  For those attributes with only a single instance per
1594            job, this index value shall be 1.  For those attributes that are
1595            a single value per document, the index value shall be the
1596            document number, starting with 1 for the first document in the
1597            job.  Jobs with only a single document shall use the index value
1598            of 1.  For those attributes that can have multiple values per
1599            job and per document, such as documentFormatIndex or
1600            documentFormatEnum, the index shall be a running index for the
1601            job as a whole, starting at 1.
1602
1603            Each job shall be identified by jmJobIndex value and each job
1604            shall be in one job set identified by jmJobSetIndex."
1605        ::= { jmAttributeEntry 2 }
1606
1607    jmAttributeValueAsInteger OBJECT-TYPE
1608        SYNTAX      Integer32(0..2147483647)
1609        MAX-ACCESS  read-only
1610        STATUS      current
1611        DESCRIPTION
1612            "The integer value of the attribute.  The value of the attribute
1613            shall be represented as an integer if the enum description
1614            JmAttributeTypeTC definition (see JmAttributeTypeTC on page 25)
1615            has the tag: 'Integer:'.
1616
1617            Depending on the enum definition, this object value may be an
1618            integer, a counter, an index, or an enum, depending on the
```

```
1619            jmAttributeTypeIndex value.  The units of this value are
1620            specified in the enum description.
1621
1622            For those attributes that are accumulating job consumption as
1623            the job is processed as specified in the JmAttributeTypeTC,
1624            shall contain the final value after the job completes
1625            processing, i.e., this value shall indicate the total usage of
1626            this resource made by the job.
1627
1628            A monitoring application is able to copy this value to a
1629            suitable longer term storage for later processing as part of an
1630            accounting system.
1631
1632            Since the agent may add attributes representing resources to
1633            this table while the job is waiting to be processed or being
1634            processed, which can be a long time before any of the resources
1635            are actually used, the agent shall set the value of the
1636            jmAttributeValueAsInteger object to 0 for resources that the job
1637            has not yet consumed.
1638
1639            Attributes for which the concept of an integer value is
1640            meaningless, such as fileName, interpreter, and
1641            physicalDeviceName, do not have the 'Integer:' tag in the
1642            JmAttributeTypeTC definition and so shall return a value of (-1)
1643            to indicate other for jmAttributeValueAsInteger."
1644        ::= { jmAttributeEntry 3 }
1645
1646    jmAttributeValueAsOctets OBJECT-TYPE
1647        SYNTAX      OCTET STRING(SIZE(0..63))
1648        MAX-ACCESS  read-only
1649        STATUS      current
1650        DESCRIPTION
1651            "The octet string value of the attribute.  The value of the
1652            attribute shall be represented as an OCTET STRING if the enum
1653            description JmAttributeTypeTC definition (see JmAttributeTypeTC
1654            on page 25) has the tag: 'Octets:'.
1655
1656            Depending on the enum definition, this object value may be a
1657            coded character set string (text) or a binary octet string, such
1658            as DateAndTime.
1659
1660            Attributes for which the concept of an octet string value is
1661            meaningless, such as pagesCompleted, do not have the tag
1662            'Octets:' in the JmAttributeTypeTC definition and so shall
1663            return a value of a zero length string for
1664            jmAttributeValueAsOctets."
1665        ::= { jmAttributeEntry 4 }
1666
1667
1668    -- Conformance Information
1669
1670    jmMIBConformance OBJECT IDENTIFIER ::= { jobmonmib 2 }
1671
```

```
-- 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, jmJobStateGroup, jmAttributeGroup
        }

        OBJECT  jmJobState
        SYNTAX      INTEGER {
                        processing(5),
                        needsAttention(7),
                        canceled(8)
                        completed(9)
        }
    DESCRIPTION
        "It is conformant for an agent to implement just these four
        states in this object.  Any additional states are conditionally
        mandatory, i.e., an agent shall represent any additional states
        that the server or device implements.  However, a client shall
        accept all of the states from an agent."

        -- OBJECT jmAttributeTypeIndex
        -- SYNTAX      INTEGER {
        --    jobState(2)
        --    numberOfInterveningJobs(5)
        --    deviceAlertCode(6)
        --    jobKOctetsRequested(30)
        --    jobKOctetsCompleted(31)
        --    impressionsRequested(35)
        --    impressionsCompleted(36)
        --    outputBinName(22)
        -- }
    -- DESCRIPTION
        --"It is conformant for an agent to implement just these 8
        -- attributes.  Any additional attributes are conditionally
        -- mandatory, i.e., an agent shall represent any additional
        -- states 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 comment 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 {
        jmGeneralJobSetName, jmGeneralJobPersistence,
        jmGeneralAttributePersistence, jmGeneralNumberOfActiveJobs,
        jmGeneralOldestActiveJobIndex,
        jmGeneralNewestActiveJobIndex }
    STATUS  current
    DESCRIPTION
        "The general group."
    ::= { jmMIBGroups 1 }

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

jmJobStateGroup OBJECT-GROUP
    OBJECTS {
        jmJobState, jmJobStateKOctetsCompleted,
        jmJobStateImpressionsCompleted, jmJobStateAssociatedValue }
    STATUS  current
    DESCRIPTION
        "The job state group."
    ::= { jmMIBGroups 3 }

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


END
```

Appendix A - Job Life Cycle

12. Job Life Cycle

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

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

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

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

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

The **jmJobState** object and the **jobState** attribute both specify the standard job states. The legal job state transitions are shown in the state transition diagram presented in

1795 Table 12-2.

Table 12-2 - Legal Job State Transition Table

	New State							
		"active" jobs						
Old state	**unknown 2**	**held 3**	**pending 4**	**processing 5**	**printing 6**	**needsAttention 7**	**canceled 8**	**completed 9**
unknown		yes	yes	yes	yes			
held			yes	yes	yes		yes	
pending		yes		yes	yes		yes	
processing		yes			yes	yes	yes	yes
printing		yes				yes	yes	yes
needsAttention		yes		yes	yes		yes	
canceled	yes							
completed	yes							

13. Bibliography

[1] The Printer MIB - RFC 1579. Also an Internet-Draft.

14. Author's Addresses

Ron Bergman
Dataproducts Corp.

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

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

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

Scott A. Isaacson
Novell, Inc.

122 E 1700 S
Provo, UT 84606

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

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

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

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

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

Other Participants:

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

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

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

15.1 Changes to version 0.71, dated 3/26/97 to make version 0.8, dated 4/4/97

1. Corresponds to the changes agreed to at the JMP meeting, 04/04/97 in Austin. Harry Lewis's changes to eliminate the **Queue** and **Completed** tables and to replace the **Job** table with the **Job ID** and **Job State** table have been incorporated.
2. Rest - TBS.

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

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

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

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

1. Added PWG agreed boiler plate Status of this Memo.
2. Updated the Abstract from Ron's comments.

3. Incorporated Ron's re-written Introduction.
4. Explained the job set concept as representing a queue within a printer or a server, if the printer or server has several or the entire set of jobs, if the printer or server has only one queue.
5. Introduced the terminology of "attribute" instead of resource, since our table represents more than just resources now, as we agreed to move many non-resource objects into it. Changed the name of the group and table from **jmResource** to **jmAttribute**.
6. Clarified that the **JmAttributeTypeTC** and **jmAttributeTable** contains information about the job, such as file name, document name, , as well as resources requested and/or consumed. Re-organized the attributes into groups of similar attributes.
7. Added more explanation about configuration 1 and 2 and added Configuration 3 as agreed to cover the case of a monitoring application that monitors a server not using SNMP while also monitoring using our MIB the printer(s) that the server controls.
8. Added more explanation of the security, internationalization, and IANA considerations.
9. Deleted the Job Set Group, since the monitoring application can find all the job sets via a Get.
10. Removed the **jmResourceUnits** object and specified the units in each **jmAttributeTypeIndex** enum. This makes it clearer what the units are and reduces the variability between agent implementations, thus making monitoring applications easier. Also cleanup the attribute names by adding the data type to the attribute name for those attributes that have more than one type that differs in the units (**Index** vs. **Name**, **Name** vs. **Enum**, **DateAndTime** vs. **TimeStamp**).
11. Added the **TimeStamp** data type as an alternative to **DateAndTime** and doubled the number of attributes that have to do with time.
12. Deleted the **JmQueuingAlgorithmTC** and **JmResourceUnitsTC** textual-conventions.
13. Added **other**(1) and **unknown**(2) to the **JmJobTypesTC** and moved the rest of the bits over.
14. Added **other**(1) to the **JmJobStateTC.**
15. Added **jobPrinting**(45) to the **JmJobStateReasonsTC** to align with IPP.
16. Move 9 objects from the **jmJobTable** to the **JmAttributeTypeTC** and **jmAttributeTable,** making them attributes: **jobAccountName**, **jobComment**, **jobSourceChannelIndex**, **physicalDeviceName**, **jobKOctetsRequested**, **jobKOctetsCompleted**, **jobSubmissionDateAndTime**, **jobSubmissionTime**, **jobStartedProcessingDateAndTime, jobStartedProcessingTime**, **jobCompletedDateAndTime**, **jobCompletedTime**. NOTE that some objects

1956 became two attributes as we have two forms of time. Also made the end of each name
1957 indicate the data type.

1958 17. Added **Requested**, **Completed**, and **CompletedCurrentCopy** forms for impressions,
1959 sheets, and pages attributes.

1960 18. Added: **other**(1), **outputBin**(9) attributes.

1961 19. Added "CPU" to **processingCPUTime** attribute.

1962 20. Added jmGeneralJobSetName so that the user could associate a name with a job set
1963 when the implementation had more than one job set. The name would typically be the
1964 queue name in such a case.

1965 21. Added **jmGeneralNumberOfJobsCompleted** and renamed
1966 jmGeneralCurrentN**u**mberOfJobs to **jmGeneralNumberOfJobsToComplete**, so that
1967 a monitoring application can find out how many jobs have completed for the
1968 **jmCompletedTable** and how many are still to be comppleted. Their sum in the total
1969 number of jobs in the **jmJobTable**.

1970 22. Clarified that **jmQueueIndex** shall be monitonically increasing which can change as
1971 new job arrive or the configuration changes.

1972 23. Added the word **Queue** to make **jmQueueJobIndex** in the Queue table.

1973 24. Clarifed that the **jmQueueJobIndex** and **jmJobIndex** shall not be 0 as required by
1974 SNMP for indexes. This gives agents that want to use the job-identifier that is
1975 generated by the system as the value for the **jmJobIndex** and **jmQueueJobIndex** a
1976 problem, if 0 is a legal value, such as in LPD.

1977 25. Clarified the distinction betwen **jmJobName** and **jmJobComment** (now jobComment
1978 attribute): jmJobName is more of a name for identificaion purposes while jobComment
1979 is free form text that often isn't present and is intended to convey anything the
1980 submitting user wanted to convey usually to him/herself.

1981 26. Clarified that -2 (unknown) shall be returned if the value of jmJobIndexNumber is
1982 unknown as in the Printer MIB convention.

1983 27. Added "**OrQueue**" to make **jmJobDeviceNameOrQueueRequested**, since some
1984 didn't know which object to use for a system in which the user specifies a queue.

1985 28. Added upper bound in **jmJobIndex** so that the MIB would compile.

1986 29. Added "**Index**" to make **jmAttributeTypeIndex** object, since this object is both a
1987 type and an index.

1988 30. Changed the name of the **jmResourceIndex** to **jmAttributeInstanceIndex**, since this
1989 index can be used for attributes that can have more than one instance per job, such as
1990 **fileName**, **documentFormat**, **outputBin**, etc.

1991 31. Clarified that the jmAttributeInstanceIndex shall be the document number for those
1992 attributes that are one to one with a document, such as **fileName**(3) and
1993 **documentName**(4).

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

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 not special prefix.

—S—