INTERNET-DRAFT R. Bergman
Dataproducts Corp.
T. Hastings (editor)
Xerox Corporation
S. Isaacson
Novell, Inc.
H. Lewis
IBM Corp.
October 2February 20, 1998

Job Monitoring MIB - V2.01.2

<draft-ietf-printmib-job-monitor-v2-0008.txt>

Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of [RFC2026]. 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."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed as http://www.ietf.org/shadow.html.

~~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).~~

This Internet-Draft expires on ~~April 2~~August 20, 1998.

Abstract

This document has been developed and approved by the Printer Working Group (PWG) as a PWG standard. It is intended to be distributed as an Informational RFC. This document provides a printer industry standard SNMP MIB for (1) monitoring the status and progress of print jobs (2) obtaining resource requirements before a job is processed, (3) monitoring resource consumption while a job is being processed and (4) collecting resource accounting data after the completion of a job. This MIB is

intended to be implemented (1) in a printer or (2) in a server that supports one or more printers. Use of the object set is not limited to printing. However, support for services other than printing is outside the scope of this Job Monitoring MIB. Future extensions to this MIB may include, but are not limited to, fax machines and scanners.

TABLE OF CONTENTS

Job Monitoring MIB

1 Introduction

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

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

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

1.1 Types of Information in the MIB

The job MIB is intended to provide the following information for the indicated Role Models in the Printer MIB[print-mib] (Appendix D - Roles of Users).

User:

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

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

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

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

Operator:

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

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

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

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

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

Capacity Planner:

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

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

Accountant:

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

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

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

1.2 Types of Job Monitoring Applications

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

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

The MIB provides a set of objects that represent a compatible subset of job and document attributes of the ISO DPA standard[iso-dpa] and the Internet Printing Protocol (IPP)[ipp-model], so that coherence is maintained between these two protocols and the information presented to end users and system operators by monitoring applications. However, the job monitoring MIB is intended to be used with printers that implement other job submitting and management protocols, such as IEEE 1284.1 (TIPSI)[tipsi], as well as with ones that do implement ISO DPA.

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

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

2 Terminology and Job Model

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

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

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

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

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

Client: The network entity that *end users* use to submit jobs to *spoolers*, *servers*, or *printers* and other *devices*, depending on the configuration, using any job submission protocol over a serial or parallel port to a directly-connected device or over the network to a networked-connected device.

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

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

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

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

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

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

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

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

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

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

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

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

Job Monitoring Application: The SNMP management application that End Users, and System Operators use to monitor jobs using SNMP. A monitor MAY be either a separate application or MAY be part of the client that also submits jobs. See "monitor".

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

Monitor: Short for Job Monitoring Application.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

2.1 System Configurations for the Job Monitoring MIB

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

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

2.1.1 Configuration 1 - client-printer

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

The job submitting client and/or monitoring application monitor jobs by communicating directly with an agent that is part of the printer. The agent in the printer SHALL keep the job in the Job Monitoring MIB as long as the job is in the printer, plus a defined time period after the job enters the completed state in which accounting programs can copy out the accounting data from the Job Monitoring MIB.

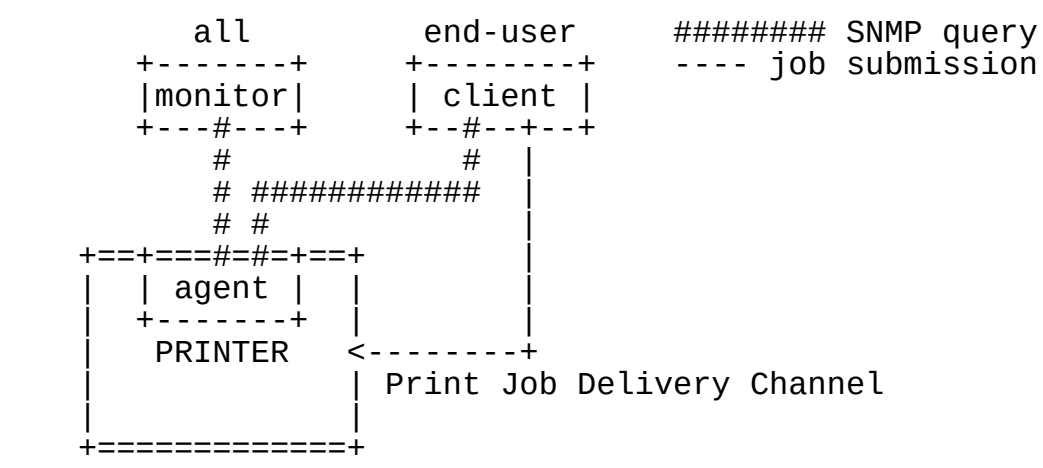

Figure 2-1 - Configuration 1 - client-printer - agent in the printer

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

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

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

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

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

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

There is no SNMP Job Monitoring MIB agent in the printer in configuration 2, at least that the client or monitor are aware. In this configuration, the agent SHALL return the current values of the objects in the Job Monitoring MIB both for jobs the server keeps and jobs that the server has submitted to the printer. The Job Monitoring MIB agent obtains the required information from the printer by a method that is beyond the scope of this document. The agent in the server SHALL keep the job in the Job Monitoring MIB in the server as long as the job is in the printer, plus a defined time period after the job enters the completed state in which accounting programs can copy out the accounting data from the Job Monitoring MIB.

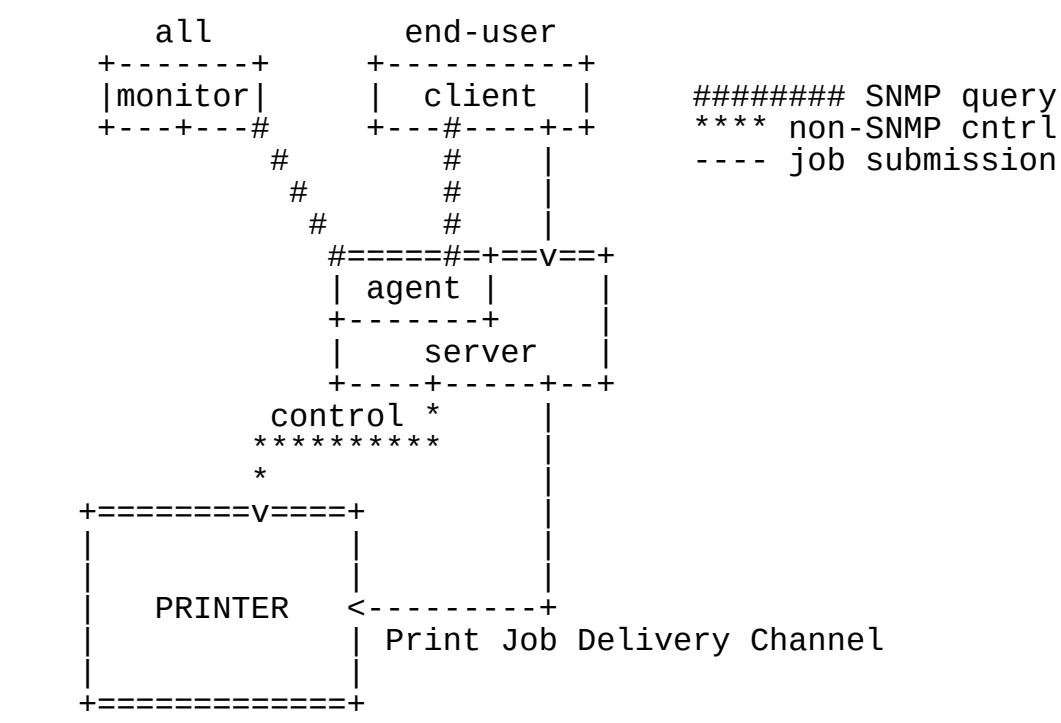

Figure 2-2 - Configuration 2 - client-server-printer - agent in the server

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

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

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

In the client-server-printer configuration 3, the client(s) submit jobs to an intermediate server by some network connection, *not* directly to the printer. That server does *not* contain a Job Monitoring MIB agent.

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

1. The server using some undefined protocol to monitor jobs in the server (that does not contain the Job Monitoring MIB) AND
2. A Job Monitoring MIB agent that is part of the printer to monitor jobs after the server passes the jobs to the printer.

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

In configuration 3, the agent (in the printer) SHALL keep the values of the objects in the Job Monitoring MIB that the agent implements updated for a job that the server has submitted to the printer. The agent SHALL obtain information about the jobs submitted to the printer from the server (either in the job submission protocol, in the document data, or by direct query of the server), in order to populate some of the objects the Job Monitoring MIB in the printer. The agent in the printer SHALL keep the job in the Job Monitoring MIB as long as the job is in the Printer, and longer in order to implement the completed state in which monitoring programs can copy out the accounting data from the Job Monitoring MIB.

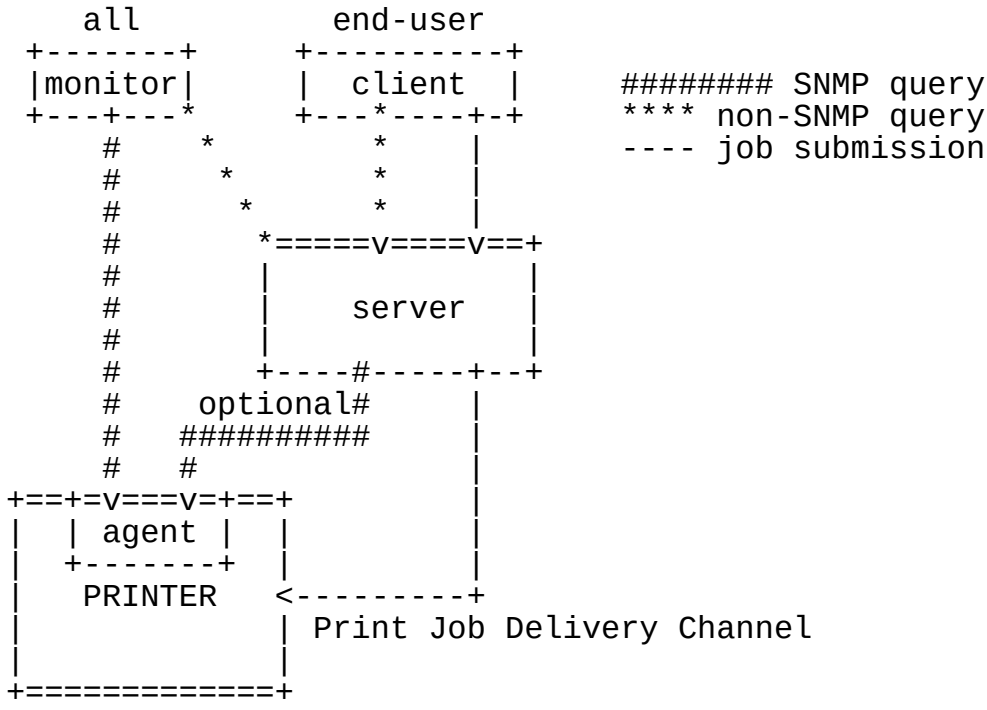

Figure 2-3 - Configuration 3 - client-server-printer - client monitors printer agent and server

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

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

3 Managed Object Usage

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

3.1 Conformance Considerations

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

3.1.1 Conformance Terminology

This specification uses the verbs: "SHALL", "SHOULD", "MAY", and "NEED NOT" to specify conformance requirements according to RFC 2119 [RFC2119~~req-words~~] as follows:

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

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

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

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

3.1.2 Agent Conformance Requirements

A conforming agent:

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

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

3.1.2.1 MIB II System Group objects

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

3.1.2.2 MIB II Interface Group objects

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

3.1.2.3 Printer MIB objects

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

3.1.3 Job Monitoring Application Conformance Requirements

A conforming job monitoring application:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

3.3 The Attribute Mechanism and the Attribute Table(s)

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

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

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

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

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

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

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

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

NOTE - The table of contents lists all the attributes in order. This order is the order of enum assignments which is the order that the SNMP GetNext operation returns attributes. Most attributes apply to all

three configurations covered by this MIB specification (see section 2.1 entitled "System Configurations for the Job Monitoring MIB"). Those attributes that apply to a particular configuration are indicated as 'Configuration *n*:' and SHALL NOT be used with other configurations.

3.3.1 Conformance of Attribute Implementation

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

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

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

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

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

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

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

for the jmAttributeValueAsOctets object. For octet string only attributes, the agent SHALL always return a '-1' value for the jmAttributeValueAsInteger object.

3.3.3 Index Value Attributes

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

3.3.4 Data Sub-types and Attribute Naming Conventions

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

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

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

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

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

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

3.3.6 Requested Objects and Attributes

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

3.3.7 Consumption Attributes

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

3.3.8 Attribute Specifications

This section specifies the job attributes.

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

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

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

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

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

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

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

NOTE: No attribute name exceeds 31 characters.

The standard attribute types are:

jmAttributeTypeIndex	Datatype
other(1),	Integer32 (-2..2147483647) AND/OR OCTET STRING(SIZE(0..63))
INTEGER: and/or OCTETS: An attribute that is not in the list and/or that has not been approved and registered with the PWG.	
+ Job State attributes (3 - 19 decimal) + + The following attributes specify the state of a job. + support bits starting: { '10'H }	
jobStateReasons2(3),	JmJobStateReasons2TC
INTEGER: Additional information about the job's current state that augments the jmJobState object. See the description under the JmJobStateReasons1TC textual-convention.	
jobStateReasons3(4),	JmJobStateReasons3TC
INTEGER: Additional information about the job's current state that augments the jmJobState object. See the description under JmJobStateReasons1TC textual-convention.	
jobStateReasons4(5),	JmJobStateReasons4TC
INTEGER: Additional information about the job's current state that augments the jmJobState object. See the description under JmJobStateReasons1TC textual-convention.	

processingMessage(6), JmUTF8StringTC (SIZE(0..63))
OCTETS: MULTI-ROW: A coded character set message that is generated by the server or device during the processing of the job as a simple form of processing log to show progress and any problems. The natural language of each value is specified by the corresponding processingMessageNaturalLangTag(7) value.

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

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

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

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

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

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

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

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

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

jobNaturalLanguageTag(9), OCTET STRING(SIZE(0..63))
OCTETS: The natural language of the job attributes supplied by the job submitter or defaulted by the server or device for the job, i.e., all objects and attributes represented by the 'JmJobStringTC' textual-convention, such as jobName, mediumRequested, etc. See Section 3.6.2, entitled 'Text supplied by the job submitter'.

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

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

        jobURI(20),                         OCTET STRING(SIZE(0..63))
            OCTETS:  MULTI-ROW:  The job's Universal Resource
            Identifier (URI) [RFC-1738].  See IPP [ipp-model] for
            example usage.

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

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

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

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

            NOTE: This attribute NEED NOT be printable characters.

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

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

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

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

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

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

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

jobServiceTypes(24), JmJobServiceTypesTC
INTEGER: Specifies the type(s) of service to which the job has been submitted (print, fax, scan, etc.). The service type is bit encoded with each job service type so that more general and arbitrary services can be created, such as services with more than one destination type, or ones with only a source or only a destination. For example, a job service might scan, faxOut, and print a single job. In this case, three bits would be set in the jobServiceTypes attribute, corresponding to the hexadecimal values: 0x8 + 0x20 + 0x4, respectively, yielding: 0x2C.

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

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

jobSourceChannelIndex(25), Integer32 (0..2147483647)
INTEGER: The index of the row in the associated Printer MIB[print-mib] of the channel which is the source of the print job.

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

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

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

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

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

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

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

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

AND/OR

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

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

The agent SHOULD return this attribute if the job has more than one document.

fileName(34), JmJobStringTC (SIZE(0..63))
OCTETS: MULTI-ROW: The coded character set file name or URI[URI-spec] of the document.

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

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

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

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

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

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

NOTE - As with all intensive attributes where multiple rows are allowed, there SHALL be only one distinct row for each distinct interpreter; there SHALL be no duplicates.

NOTE - This attribute type is intended to be used with an agent that implements the Printer MIB and SHALL not be used if the agent does not implement the Printer MIB. Such an agent SHALL use the documentFormat attribute instead.

```
1284
1285          documentFormat(38),                PrtInterpreterLangFamilyTC
1286                                             AND/OR
1287                                             OCTET STRING(SIZE(0..63))
1288              INTEGER:  MULTI-ROW:  The interpreter language family
1289              corresponding to the Printer MIB[print-mib]
1290              prtInterpreterLangFamily object, that this job
1291              requires/uses.  A document or a job MAY use more than one
1292              PDL or control language.
1293
1294              AND/OR
1295
1296              OCTETS:  MULTI-ROW:  The document format registered as a
1297              media type[iana-media-types], i.e., the name of the MIME
1298              content-type/subtype.  Examples: 'application/postscript',
1299              'application/vnd.hp-PCL', 'application/pdf', 'text/plain'
1300              (US-ASCII SHALL be assumed), 'text/plain; charset=iso-8859-
1301              1', and 'application/octet-stream'.  The IPP 'document-
1302              format' job attribute uses these same values with the same
1303              semantics.  See the IPP [ipp-model] 'mimeMediaType'
1304              attribute syntax and the document-format attribute for
1305              further examples and explanation.
1306
1307          ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
1308          + Job Parameter attributes (50 - 67 decimal)                 |
1309          +
1310          + The following attributes represent input parameters
1311          + supplied by the submitting client in the job submission
1312          + protocol.
1313          +        support bits starting: { '00000000 000020'H }       |
1314          ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
1315
1316          jobPriority(50),                   Integer32 (-2..100)
1317              INTEGER:  The priority for scheduling the job.  It is used
1318              by servers and devices that employ a priority-based
1319              scheduling algorithm.
1320
1321              A higher value specifies a higher priority.  The value 1 is
1322              defined to indicate the lowest possible priority (a job
1323              which a priority-based scheduling algorithm SHALL pass over
1324              in favor of higher priority jobs).  The value 100 is
1325              defined to indicate the highest possible priority.
1326              Priority is expected to be evenly or 'normally' distributed
1327              across this range.  The mapping of vendor-defined priority
1328              over this range is implementation-specific.  -2 indicates
1329              unknown.
```

```
1330
1331          jobProcessAfterDateAndTime(51),   DateAndTime (SNMPv2-TC)
1332              OCTETS:  The calendar date and time of day after which the
1333              job SHALL become a candidate to be scheduled for
1334              processing.  If the value of this attribute is in the
1335              future, the server SHALL set the value of the job's
1336              jmJobState object to pendingHeld and add the
1337              jobProcessAfterSpecified bit value to the job's
1338              jmJobStateReasons1 object.  When the specified date and
1339              time arrives, the server SHALL remove the
1340              jobProcessAfterSpecified bit value from the job's
1341              jmJobStateReasons1 object and, if no other reasons remain,
1342              SHALL change the job's jmJobState object to pending.
1343
1344          jobHold(52),                        JmBooleanTC
1345              INTEGER:  If the value is 'true(4)', a client has
1346              explicitly specified that the job is to be held until
1347              explicitly released.  Until the job is explicitly released
1348              by a client, the job SHALL be in the pendingHeld state with
1349              the jobHoldSpecified value in the jmJobStateReasons1
1350              attribute.
1351
1352          jobHoldUntil(53),                   JmJobStringTC (SIZE(0..63))
1353              OCTETS:  The named time period during which the job SHALL
1354              become a candidate for processing, such as 'evening',
1355              'night', 'weekend', 'second-shift', 'third-shift', etc.,
1356              (supported values configured by the system administrator).
1357              See IPP [ipp-model] for the standard keyword values.  Until
1358              that time period arrives, the job SHALL be in the
1359              pendingHeld state with the jobHoldUntilSpecified value in
1360              the jmJobStateReasons1 object.  The value 'no-hold' SHALL
1361              indicate explicitly that no time period has been specified;
1362              the absence of this attribute SHALL indicate implicitly
1363              that no time period has been specified.
1364
1365          outputBin(54),                      Integer32 (0..2147483647)
1366                                              AND/OR
1367                                              JmJobStringTC (SIZE(0..63))
1368              INTEGER:  MULTI-ROW:  The output subunit index in the
1369              Printer MIB[print-mib]
1370
1371              AND/OR
1372
1373              OCTETS:  MULTI-ROW:  the name or number (represented as
1374              ASCII digits) of the output bin to which all or part of the
1375              job is placed in.
1376
1377          sides(55),                          Integer32 (-2..2)
1378              INTEGER:  MULTI-ROW:  The number of sides, '1' or '2', that
1379              any document in this job requires/used.
```

```
1380
1381          finishing(56),                        JmFinishingTC
1382              INTEGER:  MULTI-ROW:  Type of finishing that any document
1383              in this job requires/used.
1384
1385
1386          ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
1387          + Image Quality attributes (requested and consumed) (70 - 87)
1388          +
1389          + For devices that can vary the image quality.
1390          +        support bits starting: { '00000000 00000000 02'H }
1391          ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
1392
1393          printQualityRequested(70),            JmPrintQualityTC
1394              INTEGER:  MULTI-ROW:  The print quality selection requested
1395              for a document in the job for printers that allow quality
1396              differentiation.
1397
1398          printQualityUsed(71),                 JmPrintQualityTC
1399              INTEGER:  MULTI-ROW:  The print quality selection actually
1400              used by a document in the job for printers that allow
1401              quality differentiation.
1402
1403          printerResolutionRequested(72),   JmPrinterResolutionTC
1404              OCTETS:  MULTI-ROW:  The printer resolution requested for a
1405              document in the job for printers that support resolution
1406              selection.
1407
1408          printerResolutionUsed(73),        JmPrinterResolutionTC
1409              OCTETS:  MULTI-ROW:  The printer resolution actually used
1410              by a document in the job for printers that support
1411              resolution selection.
1412
1413          tonerEcomonyRequested(74),            JmTonerEconomyTC
1414              INTEGER:  MULTI-ROW:  The toner economy selection requested
1415              for documents in the job for printers that allow toner
1416              economy differentiation.
1417
1418          tonerEcomonyUsed(75),                 JmTonerEconomyTC
1419              INTEGER:  MULTI-ROW:  The toner economy selection actually
1420              used by documents in the job for printers that allow toner
1421              economy differentiation.
1422
1423          tonerDensityRequested(76)             Integer32 (-2..100)
1424              INTEGER:  MULTI-ROW:  The toner density requested for a
1425              document in this job for devices that can vary toner
1426              density levels.  Level 1 is the lowest density and level
1427              100 is the highest density level.  Devices with a smaller
1428              range, SHALL map the 1-100 range evenly onto the
1429              implemented range.
```

```
1430
1431          tonerDensityUsed(77),              Integer32 (-2..100)
1432              INTEGER:  MULTI-ROW:  The toner density used by documents
1433              in this job for devices that can vary toner density levels.
1434              Level 1 is the lowest density and level 100 is the highest
1435              density level.  Devices with a smaller range, SHALL map the
1436              1-100 range evenly onto the implemented range.
1437
1438          ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
1439          + Job Progress attributes (requested and consumed) (90-109)
1440          +
1441          + Pairs of these attributes can be used by monitoring
1442          + applications to show an indication of relative progress
1443          + to users.  See section 3.4, entitled:
1444          + 'Monitoring Job Progress'.
1445          +    support bits starting: { '00000000 00000000 00000020'H }
1446          ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
1447
1448          jobCopiesRequested(90),            Integer32 (-2..2147483647)
1449              INTEGER:  The number of copies of the entire job that are
1450              to be produced.
1451
1452          jobCopiesCompleted(91),            Integer32 (-2..2147483647)
1453              INTEGER:  The number of copies of the entire job that have
1454              been completed so far.
1455
1456          documentCopiesRequested(92),       Integer32 (-2..2147483647)
1457              INTEGER:  The total count of the number of document copies
1458              requested for the job as a whole.  If there are documents
1459              A, B, and C, and document B is specified to produce 4
1460              copies, the number of document copies requested is 6 for
1461              the job.
1462
1463              This attribute SHALL be used only when a job has multiple
1464              documents.  The jobCopiesRequested attribute SHALL be used
1465              when the job has only one document.
1466
1467          documentCopiesCompleted(93),       Integer32 (-2..2147483647)
1468              INTEGER:  The total count of the number of document copies
1469              completed so far for the job as a whole.  If there are
1470              documents A, B, and C, and document B is specified to
1471              produce 4 copies, the number of document copies starts a 0
1472              and runs up to 6 for the job as the job processes.
1473
1474              This attribute SHALL be used only when a job has multiple
1475              documents.  The jobCopiesCompleted attribute SHALL be used
1476              when the job has only one document.
```

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

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

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

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

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

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

jobCollationType(97), JmJobCollationTypeTC
INTEGER: The type of job collation. See also Section 3.4, entitled 'Monitoring Job Progress'.

```
        ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
        + Impression attributes (110 - 129 decimal)
        +
        + See the definition of the terms 'impression', 'sheet',
        + and 'page' in Section 2.
        +
        + See also jmJobImpressionsPerCopyRequested and
        + jmJobImpressionsCompleted objects in the jmJobTable.
        + support bits starting: { '00000000 00000000 00000000 0002'H }
        ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

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

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

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

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

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

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

```
1569
1570          highlightColorImpressionsCompleted(115),
1571                                              Integer32 (-2..2147483647)
1572              INTEGER:  The number of highlight color impressions
1573              completed by the device for this job so far.  For printing,
1574              the impressions completed includes interpreting, marking,
1575              and stacking the output.  For other types of job services,
1576              the number of impressions completed includes the number of
1577              impressions processed.  Highlight color impressions are
1578              typically defined as those requiring black plus one other
1579              colorant, but this MAY vary by implementation.  In any
1580              case, the value of this attribute counts by 1 for each side
1581              that has highlight color (and the other impression counters
1582              are incremented, except
1583              fullColorImpressionsCompleted(114)).
1584
1585          ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
1586          + Page attributes (130 - 149 decimal)                          |
1587          +
1588          + See the definition of 'impression', 'sheet', and 'page'
1589          + in Section 2.
1590          + support bits starting:                                       |
1591          + { '00000000 00000000 00000000 00000000 20'H }                |
1592          ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
1593
1594          pagesRequested(130),                Integer32 (-2..2147483647)
1595              INTEGER:  The number of logical pages requested by the job
1596              to be processed.
1597
1598          pagesCompleted(131),                Integer32 (-2..2147483647)
1599              INTEGER:  The number of logical pages completed for this
1600              job so far.
1601
1602              For implementations where multiple copies are produced by
1603              the interpreter with only a single pass over the data, the
1604              final value SHALL be equal to the value of the
1605              pagesRequested object.  For implementations where multiple
1606              copies are produced by the interpreter by processing the
1607              data for each copy, the final value SHALL be a multiple of
1608              the value of the pagesRequested object.
1609
1610              NOTE - See the impressionsCompletedCurrentCopy and
1611              pagesCompletedCurrentCopy attributes for attributes that
1612              are reset on each document copy.
1613
1614              NOTE - The pagesCompleted object can be used with the
1615              pagesRequested object to provide an indication of the
1616              relative progress of the job, provided that the
1617              multiplicative factor is taken into account for some
1618              implementations of multiple copies.
```

```
1619
1620          pagesCompletedCurrentCopy(132),   Integer32 (-2..2147483647)
1621              INTEGER:  The number of logical pages completed for the
1622              current copy of the document so far.  This value SHALL be
1623              reset to 0 for each document in the job and for each
1624              document copy.
1625
1626          ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
1627          + Sheet attributes (150 - 169 decimal)                       |
1628          +
1629          + See the definition of 'impression', 'sheet', and 'page'
1630          + in Section 2.
1631          + support bits starting:                                     |
1632          + { '00000000 00000000 00000000 00000000 000002'H }          |
1633          ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
1634
1635          sheetsRequested(150),             Integer32 (-2..2147483647)
1636              INTEGER:  The total number of medium sheets requested to be
1637              produced for this job.
1638
1639              Unlike the jmJobKOctetsPerCopyRequested and
1640              jmJobImpressionsPerCopyRequested attributes, the
1641              sheetsRequested(150) attribute SHALL include the
1642              multiplicative factor contributed by the number of copies
1643              and so is the total number of sheets to be produced by the
1644              job, as opposed to the size of the document(s) submitted.
1645
1646          sheetsCompleted(151),             Integer32 (-2..2147483647)
1647              INTEGER:  The total number of medium sheets that have
1648              completed marking and stacking for the entire job so far
1649              whether those sheets have been processed on one side or on
1650              both.
1651
1652          sheetsCompletedCurrentCopy(152),  Integer32 (-2..2147483647)
1653              INTEGER:  The number of medium sheets that have completed
1654              marking and stacking for the current copy of a document in
1655              the job so far whether those sheets have been processed on
1656              one side or on both.
1657
1658              The value of this attribute SHALL be 0 before the job
1659              starts processing and SHALL be reset to 1 after the first
1660              sheet of each document and document copy in the job is
1661              processed and stacked.
1662
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+ Resources attributes (requested and consumed) (170 - 189)
+
+ Pairs of these attributes can be used by monitoring
+ applications to show an indication of relative usage to
+ users, i.e., a 'thermometer'.
+ support bits starting:
+ { '00000000 00000000 00000000 00000000 00000000 0020'H }
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

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

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

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

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

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

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

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

colorantConsumed(173), Integer32 (-2..2147483647)
AND/OR
JmJobStringTC (SIZE(0..63))

INTEGER: MULTI-ROW: The index (prtMarkerColorantIndex) in the Printer MIB[print-mib]
AND/OR
OCTETS: MULTI-ROW: the name of the colorant consumed.

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

mediumTypeConsumed(174), Integer32 (-2..2147483647)
AND
JmJobStringTC (SIZE(0..63))

INTEGER: MULTI-ROW: The number of sheets of the indicated medium type that has been consumed so far whether those sheets have been processed on one side or on both
AND
OCTETS: MULTI-ROW: the name of that medium type.

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

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

mediumSizeConsumed(175), Integer32 (-2..2147483647)
AND
JmJobStringTC (SIZE(0..63))

INTEGER: MULTI-ROW: The number of sheets of the indicated medium size that has been consumed so far whether those sheets have been processed on one side or on both
AND
OCTETS: MULTI-ROW: the name of that medium size.

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

NOTE - The size name (JmJobStringTC) values correspond to the size name values in the Printer MIB [print-mib] Appendix B. These size name values are also a subset of the keyword values defined by [ipp-model] for the 'media' Job Template attribute. Values are: 'letter', 'a', 'iso-a4', 'jis-b4', etc.

```
          ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
          + Time attributes (set by server or device) (190 - 209 decimal)
          +
          + This section of attributes are ones that are set by the
          + server or device that accepts jobs.  Two forms of time are
          + provided.  Each form is represented in a separate attribute.
          + See section 3.1.2 and section 3.1.3 for the
          + conformance requirements for time attribute for agents and
          + monitoring applications, respectively.  The two forms are:
          +
          + 'DateAndTime' is an 8 or 11 octet binary encoded year,
          + month, day, hour, minute, second, deci-second with
          + optional offset from UTC.  See SNMPv2-TC [SMIv2-TC].
          +
          + NOTE: 'DateAndTime' is not printable characters; it is
          + binary.
          +
          + 'JmTimeStampTC' is the time of day measured in the number of
          + seconds since the system was booted.
          + support bits starting:
          + { '00000000 00000000 00000000 00000000 00000000 00000002'H }
          ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

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

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

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

jobStartedProcessingTime(193),	JmTimeStampTC AND/OR DateAndTime	
INTEGER: The time AND/OR OCTETS: the date and time that the job started processing.		
jobCompletionTime(194),	JmTimeStampTC AND/OR DateAndTime	
INTEGER: The time AND/OR OCTETS: the date and time that the job entered the completed, canceled, or aborted state.		
jobProcessingCPUTime(195)	Integer32 (-2..2147483647)	
UNITS 'seconds' INTEGER: The amount of CPU time in seconds that the job has been in the processing state. If the job enters the processingStopped state, that elapsed time SHALL not be included. In other words, the jobProcessingCPUTime value SHOULD be relatively repeatable when the same job is processed again on the same device.		

3.3.9 Job State Reason bit definitions

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

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

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

3.3.9.1 JmJobStateReasons1TC specification

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

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

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

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

submissionInterrupted 0x8
The job was not completely submitted for some unforeseen reason, such as: (1) the server has crashed before the job was closed by the client, (2) the server or the document transfer method has crashed in some non-recoverable way before the document data was entirely transferred to the server, (3) the client crashed or failed to close the job before the time-out period.

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

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

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

jobProcessAfterSpecified 0x80
The value of the job's jobProcessAfterDateAndTime(51) attribute specifies a time that is still in the future. The job SHALL NOT be a candidate for processing until this reason is removed and there are no other reasons to hold the job.

resourcesAreNotReady 0x100
At least one of the resources needed by the job, such as media, fonts, resource objects, etc., is not ready on any of the physical devices for which the job is a candidate. This condition MAY be detected when the job is accepted, or subsequently while the job is pending or processing, depending on implementation.

deviceStoppedPartly 0x200
One or more, but not all, of the devices to which the job is assigned are stopped. If all of the devices are stopped (or the only device is stopped), the deviceStopped reason SHALL be used.

deviceStopped 0x400
The device(s) to which the job is assigned is (are all) stopped.

jobInterpreting 0x800
The device to which the job is assigned is interpreting the document data.

jobPrinting 0x1000
The output device to which the job is assigned is marking media. This value is useful for servers and output devices which spend a great deal of time processing (1) when no marking is happening and then want to show that marking is now happening or (2) when the job is in the process of being canceled or aborted while the job remains in the processing state, but the marking has not yet stopped so that impression or sheet counts are still increasing for the job.

jobCanceledByUser 0x2000
The job was canceled by the owner of the job, i.e., by a user whose name is the same as the value of the job's jmJobOwner object, or by some other authorized end-user, such as a member of the job owner's security group.

jobCanceledByOperator 0x4000
The job was canceled by the operator, i.e., by a user who has been authenticated as having operator privileges (whether local or remote).

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

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

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

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

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

jobCompletedSuccessfully 0x80000
The job completed successfully.

jobCompletedWithWarnings 0x100000
The job completed with warnings.

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

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

jobPaused 0x400000
The job has been indefinitely suspended by a client issuing an operation to suspend the job so that other jobs may proceed using the same devices. The client MAY issue an operation to resume the paused job at any time, in which case the agent SHALL remove the jobPaused values from the job's jmJobStateReasons1 object and the job is eventually resumed at or near the point where the job was paused.

jobInterrupted 0x800000
The job has been interrupted while processing by a client issuing an operation that specifies another job to be run instead of the current job. The server or device will automatically resume the interrupted job when the interrupting job completes.

jobRetained 0x1000000
The job is being retained by the server or device with all of the job's document data (and submitted resources, such as fonts, logos, and forms, if any). Thus a client could issue an operation to the server or device to either (1) re-do the job (or a copy of the job) on the same server or device or (2) resubmit the job to another server or device. When a client could no longer re-do/resubmit the job, such as after the document data has been discarded, the agent SHALL remove the jobRetained value from the jmJobStateReasons1 object.

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

3.3.9.2 JmJobStateReasons2TC specification

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

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

deletedByAdministrator 0x2
: The administrator has deleted the job.

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

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

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

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

devicesNeedAttentionTimeOut 0x40
: One or more document transforms that the job is using needs human intervention in order for the job to make progress, but the human intervention did not occur within the site-settable time-out value.

needsKeyOperatorTimeOut 0x80
: One or more devices or document transforms that the job is using need a specially trained operator (who may need a key to unlock the device and gain access) in order for the job to make progress, but the key operator intervention did not occur within the site-settable time-out value.

jobStartWaitTimeOut 0x100
The server/device has stopped the job at the beginning of processing to await human action, such as installing a special cartridge or special non-standard media, but the job was not resumed within the site-settable time-out value and the server/device has transitioned the job to the pendingHeld state.

jobEndWaitTimeOut 0x200
The server/device has stopped the job at the end of processing to await human action, such as removing a special cartridge or restoring standard media, but the job was not resumed within the site-settable time-out value and the server/device has transitioned the job to the completed state.

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

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

connectingToDeviceTimeOut 0x1000
The server is attempting to connect to one or more devices which may be dial-up, polled, or queued, and so may be busy with traffic from other systems, but server was unable to connect to the device within the site-settable time-out value.

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

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

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

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

jobPasswordWait 0x20000
The server/device has selected the job to be next to process, but instead of assigning resources and starting the job processing, the server/device has transitioned the job to the pendingHeld state to await entry of a password (and dispatched another job, if there is one).

validating 0x40000
The server/device is validating the job *after* accepting the job.

queueHeld 0x80000
The operator has held the entire job set or queue.

jobProofWait 0x100000
The job has produced a single proof copy and is in the pendingHeld state waiting for the requester to issue an operation to release the job to print normally, obeying any job and document copy attributes that were originally submitted.

heldForDiagnostics 0x200000
The system is running intrusive diagnostics, so that all jobs are being held.

noSpaceOnServer 0x800000
There is no room on the server to store all of the job.

pinRequired 0x1000000
The System Administrator settable device policy is (1) to require PINs, and (2) to hold jobs that do not have a pin supplied as an input parameter when the job was created.

exceededAccountLimit 0x2000000
The account for which this job is drawn has exceeded its limit. This condition SHOULD be detected before the job is scheduled so that the user does not wait until his/her job is scheduled only to find that the account is overdrawn. This condition MAY also occur while the job is processing either as processing begins or part way through processing.

heldForRetry 0x4000000
The job encountered some errors that the server/device could not recover from with its normal retry procedures, but the error might not be encountered if the job is processed again in the future. Example cases are phone number busy or remote file system in-accessible. For such a situation, the server/device SHALL transition the job from the processing to the pendingHeld, rather than to the aborted state.

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

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

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

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

badJob 0x40000000
This job was aborted by the system because this job has a major problem, such as an ill-formed PDL; the spooler SHOULD not even try another device.

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

3.3.9.3 JmJobStateReasons3TC specification

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

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

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

3.3.9.4 JmJobStateReasons4TC specification

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

None defined at this time.

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

3.4 Monitoring Job Progress

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

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

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

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

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

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

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

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

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

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

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

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

Job Collation Type = uncollatedSheets(3)

jmJobImpressionsCompleted	ImpressionsCompletedCurrentCopy	sheetCompletedCopyNumber	sheetCompletedDocumentNumber
0	0	0	0
1	1	1	1
2	1	2	1
3	1	3	1
4	2	1	1
5	2	2	1
6	2	3	1
7	3	1	1
8	3	2	1
9	3	3	1
10	1	1	2
11	1	2	2
12	1	3	2
13	2	1	2
14	2	2	2
15	2	3	2
16	3	1	2
17	3	2	2
18	3	3	2

Job Collation Type = collatedDocuments(4)

JmJobImpressionsCompleted	ImpressionsCompletedCurrentCopy	sheetCompletedCopyNumber	sheetCompletedDocumentNumber
0	0	0	0
1	1	1	1
2	2	1	1
3	3	1	1
4	1	1	2
5	2	1	2
6	3	1	2
7	1	2	1
8	2	2	1
9	3	2	1
10	1	2	2
11	2	2	2
12	3	2	2
13	1	3	1
14	2	3	1
15	3	3	1
16	1	3	2
17	2	3	2
18	3	3	2

Job Collation Type = uncollatedDocuments(5)

jmJobImpressionsCompleted	ImpressionsCompletedCurrentCopy	sheetCompletedCopyNumber	sheetCompletedDocumentNumber
0	0	0	0
1	1	1	1
2	2	1	1
3	3	1	1
4	1	2	1
5	2	2	1
6	3	2	1
7	1	3	1
8	2	3	1
9	3	3	1
10	1	1	2
11	2	1	2
12	3	1	2
13	1	2	2
14	2	2	2
15	3	2	2
16	1	3	2
17	2	3	2
18	3	3	2

3.5 Job Identification

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

The server/device-side identifier, called the jmJobIndex object, SHALL be assigned by the SNMP Job Monitoring MIB agent when the server or device accepts the jobs from submitting clients. The jmJobIndex object allows the interested party to obtain all objects desired that relate

to a particular job. See Section 3.2, entitled 'The Job Tables and the Oldest Active and Newest Active Indexes' for the specification of how the agent SHALL assign the jmJobIndex values.

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

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

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

3.5.1 The Job Submission ID specifications

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

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

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

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

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

Format Letter	Description
'0'	Job Owner generated by the server/device octets 2-40: The last 39 bytes of the jmJobOwner object. octets 41-48: The US-ASCII 8-decimal-digit sequential number assigned by the agent. This format is reserved for agents. NOTE - Clients wishing to use a job submission ID that incorporates the job owner, SHALL use format '8', not format '0'.
'1'	Job Name octets 2-40: The last 39 bytes of the jobName attribute. octets 41-48: The US-ASCII 8-decimal-digit random number assigned by the client. This format is reserved for clients.
'2'	Client MAC address octets 2-40: The client MAC address: in hexadecimal with each nibble of the 6 octet address being '0'-'9' or 'A' - 'F' (uppercase only). Most significant octet first. octets 41-48: The US-ASCII 8-decimal-digit sequential number assigned by the client. This format is reserved for clients.
'3'	Client URL octets 2-40: The last 39 bytes of the client URL [URI-spec]. octets 41-48: The US-ASCII 8-decimal-digit sequential number assigned by the client. This format is reserved for clients.
'4'	Job URI octets 2-40: The last 39 bytes of the URI [URI-spec] assigned by the server or device to the job when the job was submitted for processing. octets 41-48: The US-ASCII 8-decimal-digit sequential number assigned by the agent. This format is reserved for agents.
'5'	POSIX User Number octets 2-40: The last 39 bytes of a user number, such as POSIX user number. octets 41-48: The US-ASCII 8-decimal-digit sequential number assigned by the client.

This format is reserved for clients.

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

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

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

'9' Host Name
octets 2-40: The last 39 bytes of the host name with trailing SPACES that submitted the job to this server/device using a protocol, such as LPD [RFC-1179] which includes the host name in the job submission protocol.
octets 41-48: The US-ASCII 8-decimal-digit leading zero representation of the job id generated by the submitting server (configuration 3) or the client (configuration 1 and 2), such as in the LPD protocol.
This format is reserved for clients.

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

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

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

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

'E' IPDS on the MVS or VSE platform

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

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

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

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

3.6 Internationalization Considerations

This section describes the internationalization considerations included in this MIB.

3.6.1 Text generated by the server or device

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

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

The character encoding scheme for representing these objects and attributes SHALL be UTF-8 as REQUIRED ~~recommended~~ by RFC 2277 ~~2130~~ [RFC2277 ~~2130~~] ~~and the "IETF Policy on Character Sets and Language" [char-set policy]~~. The 'JmUTF8StringTC' textual convention is used to indicate UTF-8 text strings.

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

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

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

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

3.6.2 Text supplied by the job submitter

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

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

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

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

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

The natural language for attributes represented by the textual-convention JmJobStringTC is identified either (1) by the jobNaturalLanguageTag(9) attribute or is keywords in US-English (as in IPP). A monitoring application SHOULD attempt to localize keywords into the language of the user by means of some lookup mechanism. If the keyword value is not known to the monitoring application, the monitoring application SHOULD assume that the value is in the natural language specified by the job's jobNaturalLanguageTag(9) attribute and SHOULD present the value to its user as is. The

jobNaturalLanguageTag(9) attribute value SHALL have the same syntax and semantics as the processingMessageNaturalLangTag(7) attribute, except that the jobNaturalLanguageTag(9) attribute identifies the natural language of attributes supplied by the job submitter instead of the natural language of the processingMessage(6) attribute. See Section 3.6.1.

3.6.3 'DateAndTime' for representing the date and time

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

3.7 IANA and PWG Registration Considerations

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

1. printer-language-numbers - used as enums in the documentFormat(38) attribute
2. media-types - uses as keywords in the documentFormat(38) attribute
3. character-sets - used as enums in the jobCodedCharSet(8) attribute

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

3.7.1 PWG Registration of enums

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

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

3.7.1.1 Type 1 enumerations

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

There are no type 1 enums in the current draft.

3.7.1.2 Type 2 enumerations

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

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

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

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

3.7.1.3 Type 3 enumeration

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

There are no type 3 enums in the current draft.

3.7.2 PWG Registration of type 2 bit values

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

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

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

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

3.7.3 PWG Registration of Job Submission Id Formats

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

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

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

3.8 Security Considerations

3.8.1 Read-Write objects

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

3.8.2 Read-Only Objects In Other User's Jobs

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

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

3.9 Notifications

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

4 MIB specification

The following pages constitute the actual Job Monitoring MIB.

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

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

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

jobmonMIB MODULE-IDENTITY
    LAST-UPDATED "990220981002 0000Z"
    ORGANIZATION "Printer Working Group (PWG)"
    CONTACT-INFO
        "Tom Hastings
        Postal:  Xerox Corp.
                 Mail stop ESAE-231
                 701 S. Aviation Blvd.
                 El Segundo, CA 90245

        Tel:     (301)333-6413
        Fax:     (301)333-5514
        E-mail:  hastings@cp10.es.xerox.com

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

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

            http://www.pwg.org/

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

        Version: 2.01.2"
    ::= { enterprises pwg(2699)  mibs(1)  jobmonMIB(1) }
```

```
2801
2802
2803   -- Textual conventions for this MIB module
2804
2805   JmUTF8StringTC ::= TEXTUAL-CONVENTION
2806       DISPLAY-HINT "255a"
2807       STATUS       current
2808       DESCRIPTION
2809           "To facilitate internationalization, this TC represents
2810           information taken from the ISO/IEC IS 10646-1 character set,
2811           encoded as an octet string using the UTF-8 character encoding
2812           scheme.
2813
2814           See section 3.6.1, entitled: 'Text generated by the server or
2815           device'."
2816       SYNTAX       OCTET STRING (SIZE (0..63))
2817
2818
2819
2820
2821   JmJobStringTC ::= TEXTUAL-CONVENTION
2822       STATUS       current
2823       DESCRIPTION
2824           "To facilitate internationalization, this TC represents
2825           information using any coded character set registered by IANA as
2826           specified in section 3.7.  While it is recommended that the
2827           coded character set be UTF-8 [UTF-8], the actual coded
2828           character set SHALL be indicated by the value of the
2829           jobCodedCharSet(8) attribute for the job.
2830
2831           See section 3.6.2, entitled: 'Text supplied by the job
2832           submitter'."
2833       SYNTAX       OCTET STRING (SIZE (0..63))
2834
2835
2836
2837
2838   JmNaturalLanguageTagTC  ::= TEXTUAL-CONVENTION
2839       STATUS       current
2840       DESCRIPTION
2841           "An IETF RFC 1766-compliant 'language tag', with zero or more
2842           sub-tags that identify a natural language.  While RFC 1766
2843           specifies that the US-ASCII values are case-insensitive, this
2844           MIB specification requires that all characters SHALL be lower
2845           case in order to simplify comparing by management applications.
2846
2847           See section 3.6.1, entitled: 'Text generated by the server or
2848           device' and section 3.6.2, entitled: 'Text supplied by the job
2849           submitter'."
2850       SYNTAX       OCTET STRING (SIZE (0..63))
```

```
2851
2852
2853   JmTimeStampTC ::= TEXTUAL-CONVENTION
2854       STATUS      current
2855       DESCRIPTION
2856           "The simple time at which an event took place.  The units are
2857           in seconds since the system was booted.
2858
2859           NOTE - JmTimeStampTC is defined in units of seconds, rather
2860           than 100ths of seconds, so as to be simpler for agents to
2861           implement (even if they have to implement the 100ths of a
2862           second to comply with implementing sysUpTime in MIB-II[mib-
2863           II].)
2864
2865           NOTE - JmTimeStampTC is defined as an Integer32 so that it can
2866           be used as a value of an attribute, i.e., as a value of the
2867           jmAttributeValueAsInteger object.  The TimeStamp textual-
2868           convention defined in SNMPv2-TC [SMIv2-TC] is defined as an
2869           APPLICATION 3 IMPLICIT INTEGER tag, not an Integer32 which is
2870           defined in SNMPv2-SMI [SMIv2-TC] as UNIVERSAL 2 IMPLICIT
2871           INTEGER, so cannot be used in this MIB as one of the values of
2872           jmAttributeValueAsInteger."
2873       SYNTAX      INTEGER (0..2147483647)
2874
2875
2876
2877
2878   JmJobSourcePlatformTypeTC ::= TEXTUAL-CONVENTION
2879       STATUS      current
2880       DESCRIPTION
2881           "The source platform type that can submit jobs to servers or
2882           devices in any of the 3 configurations.
2883
2884           This is a type 2 enumeration.  See Section 3.7.1.2.  See also
2885           IANA operating-system-names registry."
2886       SYNTAX      INTEGER {
                        other(1),
                        unknown(2),
                        sptUNIX(3),           -- UNIX
                        sptOS2(4),            -- OS/2
                        sptPCDOS(5),          -- DOS
                        sptNT(6),             -- NT
                        sptMVS(7),            -- MVS
                        sptVM(8),             -- VM
                        sptOS400(9),          -- OS/400
                        sptVMS(10),           -- VMS
                        sptWindows(11),       -- Windows
                        sptNetWare(12)        -- NetWare
2887       }
```

```
2888
2889
2890   JmFinishingTC ::= TEXTUAL-CONVENTION
2891       STATUS      current
2892       DESCRIPTION
2893           "The type of finishing operation.
2894
2895           These values are the same as the enum values of the IPP
2896           'finishings' attribute.  See Section 3.7.1.2.
2897
2898           other(1),
2899               Some other finishing operation besides one of the specified
2900               or registered values.
2901
2902           unknown(2),
2903               The finishing is unknown.
2904
2905           none(3),
2906               Perform no finishing.
2907
2908           staple(4),
2909               Bind the document(s) with one or more staples. The exact
2910               number and placement of the staples is site-defined.
2911
2912           punch(5),
2913               Holes are required in the finished document. The exact
2914               number and placement of the holes is site-defined.  The
2915               punch specification MAY be satisfied (in a site- and
2916               implementation-specific manner) either by
2917               drilling/punching, or by substituting pre-drilled media.
2918
2919           cover(6),
2920               Select a non-printed (or pre-printed) cover for the
2921               document. This does not supplant the specification of a
2922               printed cover (on cover stock medium) by the document
2923               itself.
2924
2925           bind(7)
2926               Binding is to be applied to the document; the type and
2927               placement of the binding is product-specific.
2928
2929           This is a type 2 enumeration.  See Section 3.7.1.2."
2930       SYNTAX      INTEGER {
2931           other(1),
2932           unknown(2),
2933           none(3),
2934           staple(4),
2935           punch(5),
2936           cover(6),
2937           bind(7)
2938       }
```

```
2939
2940
2941   JmPrintQualityTC ::= TEXTUAL-CONVENTION
2942       STATUS      current
2943       DESCRIPTION
2944           "Print quality settings.
2945
2946           These values are the same as the enum values of the IPP 'print-
2947           quality' attribute.  See Section 3.7.1.2.
2948
2949           This is a type 2 enumeration.  See Section 3.7.1.2."
2950       SYNTAX      INTEGER {
                        other(1),     -- Not one of the specified or registered
                                      -- values.
                        unknown(2),   -- The actual value is unknown.
                        draft(3),     -- Lowest quality available on the printer.
                        normal(4),    -- Normal or intermediate quality on the
                                      -- printer.
                        high(5)       -- Highest quality available on the printer.
2951       }
2952
2953
2954
2955   JmPrinterResolutionTC ::= TEXTUAL-CONVENTION
2956       STATUS      current
2957       DESCRIPTION
2958           "Printer resolutions.
2959
2960           Nine octets consisting of two 4-octet SIGNED-INTEGERs followed
2961           by a SIGNED-BYTE.  The values are the same as those specified
2962           in the Printer MIB [printmib]. The first SIGNED-INTEGER
2963           contains the value of prtMarkerAddressabilityXFeedDir.  The
2964           second SIGNED-INTEGER contains the value of
2965           prtMarkerAddressabilityFeedDir.  The SIGNED-BYTE contains the
2966           value of prtMarkerAddressabilityUnit.
2967
2968           Note: the latter value is either 3 (tenThousandsOfInches) or 4
2969           (micrometers) and the addressability is in 10,000 units of
2970           measure. Thus the SIGNED-INTEGERs represent integral values in
2971           either dots-per-inch or dots-per-centimeter.
2972
2973           The syntax is the same as the IPP 'printer-resolution'
2974           attribute.  See Section 3.7.1.2."
2975       SYNTAX      OCTET STRING (SIZE(9))
2976
2977
```

```
2978
2979
2980   JmTonerEconomyTC ::= TEXTUAL-CONVENTION
2981       STATUS      current
2982       DESCRIPTION
2983           "Toner economy settings.
2984
2985           This is a type 2 enumeration.  See Section 3.7.1.2."
2986       SYNTAX      INTEGER {
                        unknown(2),    --  unknown.
                        off(3),        --  Off.  Normal.  Use full toner.
                        on(4)          --  On.  Use less toner than normal.
2987       }
2988
2989
2990
2991   JmBooleanTC ::= TEXTUAL-CONVENTION
2992       STATUS      current
2993       DESCRIPTION
2994           "Boolean true or false value.
2995
2996           This is a type 2 enumeration.  See Section 3.7.1.2."
2997       SYNTAX      INTEGER {
                        unknown(2),    --  unknown.
                        false(3),      --  FALSE.
                        true(4)        --  TRUE.
2998       }
2999
3000
3001
3002   JmMediumTypeTC ::= TEXTUAL-CONVENTION
3003       STATUS      current
3004       DESCRIPTION
3005           "Identifies the type of medium.
3006
3007           other(1),
3008               The type is neither one of the values listed in this
3009               specification nor a registered value.
3010
3011           unknown(2),
3012               The type is not known.
3013
3014           stationery(3),
3015               Separately cut sheets of an opaque material.
3016
3017           transparency(4),
3018               Separately cut sheets of a transparent material.
3019
3020           envelope(5),
3021               Envelopes that can be used for conventional mailing
3022               purposes.
```

```
3023
3024            envelopePlain(6),
3025                Envelopes that are not preprinted and have no windows.
3026
3027            envelopeWindow(7),
3028                Envelopes that have windows for addressing purposes.
3029
3030            continuousLong(8),
3031                Continuously connected sheets of an opaque material
3032                connected along the long edge.
3033
3034            continuousShort(9),
3035                Continuously connected sheets of an opaque material
3036                connected along the short edge.
3037
3038            tabStock(10),
3039                Media with tabs.
3040
3041            multiPartForm(11),
3042                Form medium composed of multiple layers not pre-attached to
3043                one another;  each sheet MAY be drawn separately from an
3044                input source.
3045
3046            labels(12),
3047                Label-stock.
3048
3049            multiLayer(13)
3050                Form medium composed of multiple layers which are pre-
3051                attached to one another, e.g. for use with impact printers.
3052
3053            This is a type 2 enumeration.  See Section 3.7.1.2.  These enum
3054            values correspond to the keyword name strings of the
3055            prtInputMediaType object in the Printer MIB [print-mib].  There
3056            is no printer description attribute in IPP/1.0 that represents
3057            these values."
3058        SYNTAX      INTEGER {
3059            other(1),
3060            unknown(2),
3061            stationery(3),
3062            transparency(4),
3063            envelope(5),
3064            envelopePlain(6),
3065            envelopeWindow(7),
3066            continuousLong(8),
3067            continuousShort(9),
3068            tabStock(10),
3069            multiPartForm(11),
3070            labels(12),
3071            multiLayer(13)
3072        }
3073
```

```
3074
3075
3076   JmJobCollationTypeTC ::= TEXTUAL-CONVENTION
3077       STATUS      current
3078       DESCRIPTION
3079           "This value is the type of job collation.  Implementations that
3080           don't support multiple documents or don't support multiple
3081           copies SHALL NOT support the uncollatedDocuments(5) value.
3082
3083           This is a type 2 enumeration.  See Section 3.7.1.2. See also
3084           Section 3.4, entitled 'Monitoring Job Progress'."
3085       SYNTAX      INTEGER {
3086           other(1),
3087           unknown(2),
3088           uncollatedSheets(3),     -- sheets within each document copy
3089                                    -- are not collated: 1 1 ..., 2 2 ...,
3090                                    -- No corresponding value of IPP
3091                                    -- "multiple-document-handling"
3092           collatedDocuments(4),    -- internal collated sheets,
3093                                    -- documents: A, B, A, B, ...
3094                                    -- Corresponds to IPP "multiple-
3095                                    -- document-handling"='separate-
3096                                    -- documents-collated-copies'
3097           uncollatedDocuments(5)   -- internal collated sheets,
3098                                    -- documents: A, A, ..., B, B, ...
3099                                    -- Corresponds to IPP "multiple-
3100                                    -- document-handling"='separate-
3101                                    -- documents-uncollated-copies'
3102       }
3103
3104
3105   JmJobSubmissionIDTypeTC ::= TEXTUAL-CONVENTION
3106       STATUS      current
3107       DESCRIPTION
3108           "Identifies the format type of a job submission ID.
3109
3110           Each job submission ID is a fixed-length, 48-octet printable
3111           US-ASCII [US-ASCII] coded character string containing no
3112           control characters, consisting of the fields defined in section
3113           3.5.1.~~following fields:~~
3114
3115           ~~- octet  1:  The format letter identifying the format.  The US-~~
3116               ~~ASCII characters '0-9', 'A-Z', and 'a-z' are assigned in~~
3117               ~~order giving 62 possible formats.~~
3118           ~~- octets 2-40:  A 39-character, US-ASCII trailing SPACE filled~~
3119               ~~field specified by the format letter, if the data is less~~
3120               ~~than 39 ASCII characters.~~
3121           ~~- octets 41-48:  A sequential or random US-ASCII number to make~~
3122               ~~the ID quasi-unique.~~
3123
3124           ~~If the client does not supply a job submission ID in the job~~
3125           ~~submission protocol, then the agent SHALL assign a job~~
```

submission ID using any of the standard formats that are reserved for the agent. Clients SHALL not use formats that are reserved for agents and agents SHALL NOT use formats that are reserved for clients, in order to reduce conflicts in ID generation. See the description for which formats are reserved for clients or for agents.

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

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

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

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

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

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

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

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

~~This format is reserved for agents.~~

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

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

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

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

~~'9' Host Name~~
~~octets 2-40: The last 39 bytes of the host name with trailing SPACES that submitted the job to this server/device using a protocol, such as LPD [RFC-1179] which includes the host name in the job submission protocol.~~
~~octets 41-48: The US-ASCII 8-decimal-digit leading zero representation of the job id generated by the submitting server (configuration 3) or the client (configuration 1 and 2), such as in the LPD protocol.~~
~~This format is reserved for clients.~~

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

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

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

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

~~'E' IPDS on the MVS or VSE platform~~

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

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

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

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

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

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

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

Figure 4 - Normal Job State Transitions

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

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

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

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

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

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

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

processing(5),
One or more of:

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

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

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

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

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

processingStopped(6),
The job has stopped while processing for any number of reasons and will return to the processing state as soon as the reasons are no longer present.

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

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

```
3418
3419            canceled(7),
3420                A client has canceled the job and the server or device has
3421                completed canceling the job AND all MIB objects and
3422                attributes have reached their final values for the job.
3423                While the server or device is canceling the job, the job's
3424                jmJobStateReasons1 object SHOULD contain the
3425                processingToStopPoint value and one of the canceledByUser,
3426                canceledByOperator, or canceledAtDevice values.  The
3427                canceledByUser, canceledByOperator, or canceledAtDevice
3428                values remain while the job is in the canceled state.
3429
3430            aborted(8),
3431                The job has been aborted by the system, usually while the
3432                job was in the processing or processingStopped state and
3433                the server or device has completed aborting the job AND all
3434                MIB objects and attributes have reached their final values
3435                for the job.  While the server or device is aborting the
3436                job, the job's jmJobStateReasons1 object MAY contain the
3437                processingToStopPoint and abortedBySystem values.  If
3438                implemented, the abortedBySystem value SHALL remain while
3439                the job is in the aborted state.
3440
3441            completed(9)
3442                The job has completed successfully or with warnings or
3443                errors after processing and all of the media have been
3444                successfully stacked in the appropriate output bin(s) AND
3445                all MIB objects and attributes have reached their final
3446                values for the job.  The job's jmJobStateReasons1 object
3447                SHOULD contain one of: completedSuccessfully,
3448                completedWithWarnings, or completedWithErrors values.
3449
3450            This is a type 2 enumeration.  See Section 3.7.1.2."
3451        SYNTAX      INTEGER {
3452            unknown(2),
3453            pending(3),
3454            pendingHeld(4),
3455            processing(5),
3456            processingStopped(6),
3457            canceled(7),
3458            aborted(8),
3459            completed(9)
3460        }
```

```
3461
3462
3463   JmAttributeTypeTC ::= TEXTUAL-CONVENTION
3464       STATUS      current
3465       DESCRIPTION
3466           "The type of the attribute which identifies the attribute.
3467
3468           NOTE - The enum assignments are grouped logically with values
3469           assigned in groups of 20, so that additional values may be
3470           registered in the future and assigned a value that is part of
3471           their logical grouping.
3472
3473           Values in the range 2**30 to 2**31-1 are reserved for private
3474           or experimental usage.  This range corresponds to the same
3475           range reserved in IPP.  Implementers are warned that use of
3476           such values may conflict with other implementations.
3477           Implementers are encouraged to request registration of enum
3478           values following the procedures in Section 3.7.1.
3479
3480           See Section 3.2 entitled 'The Attribute Mechanism' for a
3481           description of this textual-convention and its use in the
3482           jmAttributeTable.  See Section 3.3.8 for the specification of
3483           each attribute.  The comment(s) after each enum assignment
3484           specifies the data type(s) of the attribute.
3485
3486           This is a type 2 enumeration.  See Section 3.7.1.2."
3487
3488       SYNTAX      INTEGER {
3489           other(1),                          -- Integer32 (-2..2147483647)
3490                                              -- AND/OR
3491                                              -- OCTET STRING(SIZE(0..63))
3492
3493           -- Job State attributes:
3494           jobStateReasons2(3),               -- JmJobStateReasons2TC
3495           jobStateReasons3(4),               -- JmJobStateReasons3TC
3496           jobStateReasons4(5),               -- JmJobStateReasons4TC
3497           processingMessage(6),              -- JmUTF8StringTC (SIZE(0..63))
3498           processingMessageNaturalLangTag(7),
3499                                              -- OCTET STRING(SIZE(0..63))
3500           jobCodedCharSet(8),                -- CodedCharSet
3501           jobNaturalLanguageTag(9),          -- OCTET STRING(SIZE(0..63))
3502
```

```
3503          -- Job Identification attributes:
3504          jobURI(20),                       -- OCTET STRING(SIZE(0..63))
3505          jobAccountName(21),               -- OCTET STRING(SIZE(0..63))
3506          serverAssignedJobName(22),        -- JmJobStringTC (SIZE(0..63))
3507          jobName(23),                      -- JmJobStringTC (SIZE(0..63))
3508          jobServiceTypes(24),              -- JmJobServiceTypesTC
3509          jobSourceChannelIndex(25),        -- Integer32 (0..2147483647)
3510          jobSourcePlatformType(26),        -- JmJobSourcePlatformTypeTC
3511          submittingServerName(27),         -- JmJobStringTC (SIZE(0..63))
3512          submittingApplicationName(28),    -- JmJobStringTC (SIZE(0..63))
3513          jobOriginatingHost(29),           -- JmJobStringTC (SIZE(0..63))
3514          deviceNameRequested(30),          -- JmJobStringTC (SIZE(0..63))
3515          queueNameRequested(31),           -- JmJobStringTC (SIZE(0..63))
3516          physicalDevice(32),               -- hrDeviceIndex
3517                                            -- AND/OR
3518                                            -- JmUTF8StringTC (SIZE(0..63))
3519          numberOfDocuments(33),            -- Integer32 (-2..2147483647)
3520          fileName(34),                     -- JmJobStringTC (SIZE(0..63))
3521          documentName(35),                 -- JmJobStringTC (SIZE(0..63))
3522          jobComment(36),                   -- JmJobStringTC (SIZE(0..63))
3523          documentFormatIndex(37),          -- Integer32 (0..2147483647)
3524          documentFormat(38),               -- PrtInterpreterLangFamilyTC
3525                                            -- AND/OR
3526                                            -- OCTET STRING(SIZE(0..63))
3527
3528          -- Job Parameter attributes:
3529          jobPriority(50),                  -- Integer32 (-2..100)
3530          jobProcessAfterDateAndTime(51),   -- DateAndTime (SNMPv2-TC)
3531          jobHold(52),                      -- JmBooleanTC
3532          jobHoldUntil(53),                 -- JmJobStringTC (SIZE(0..63))
3533          outputBin(54),                    -- Integer32 (0..2147483647)
3534                                            -- AND/OR
3535                                            -- JmJobStringTC (SIZE(0..63))
3536          sides(55),                        -- Integer32 (-2..2)
3537          finishing(56),                    -- JmFinishingTC
3538
3539          -- Image Quality attributes:
3540          printQualityRequested(70),        -- JmPrintQualityTC
3541          printQualityUsed(71),             -- JmPrintQualityTC
3542          printerResolutionRequested(72),   -- JmPrinterResolutionTC
3543          printerResolutionUsed(73),        -- JmPrinterResolutionTC
3544          tonerEcomonyRequested(74),        -- JmTonerEconomyTC
3545          tonerEcomonyUsed(75),             -- JmTonerEconomyTC
3546          tonerDensityRequested(76),        -- Integer32 (-2..100)
3547          tonerDensityUsed(77),             -- Integer32 (-2..100)
3548
```

```
3549          -- Job Progress attributes:
3550          jobCopiesRequested(90),          -- Integer32 (-2..2147483647)
3551          jobCopiesCompleted(91),          -- Integer32 (-2..2147483647)
3552          documentCopiesRequested(92),     -- Integer32 (-2..2147483647)
3553          documentCopiesCompleted(93),     -- Integer32 (-2..2147483647)
3554          jobKOctetsTransferred(94),       -- Integer32 (-2..2147483647)
3555          sheetCompletedCopyNumber(95),    -- Integer32 (-2..2147483647)
3556          sheetCompletedDocumentNumber(96),
3557                                           -- Integer32 (-2..2147483647)
3558          jobCollationType(97),            -- JmJobCollationTypeTC
3559
3560          -- Impression attributes:
3561          impressionsSpooled(110),         -- Integer32 (-2..2147483647)
3562          impressionsSentToDevice(111),    -- Integer32 (-2..2147483647)
3563          impressionsInterpreted(112),     -- Integer32 (-2..2147483647)
3564          impressionsCompletedCurrentCopy(113),
3565                                           -- Integer32 (-2..2147483647)
3566          fullColorImpressionsCompleted(114),
3567                                           -- Integer32 (-2..2147483647)
3568          highlightColorImpressionsCompleted(115),
3569                                           -- Integer32 (-2..2147483647)
3570
3571          -- Page attributes:
3572          pagesRequested(130),             -- Integer32 (-2..2147483647)
3573          pagesCompleted(131),             -- Integer32 (-2..2147483647)
3574          pagesCompletedCurrentCopy(132),  -- Integer32 (-2..2147483647)
3575
3576          -- Sheet attributes:
3577          sheetsRequested(150),            -- Integer32 (-2..2147483647)
3578          sheetsCompleted(151),            -- Integer32 (-2..2147483647)
3579          sheetsCompletedCurrentCopy(152),-- Integer32 (-2..2147483647)
3580
3581          -- Resource attributes:
3582          mediumRequested(170),            -- JmMediumTypeTC
3583                                           -- AND/OR
3584                                           -- JmJobStringTC (SIZE(0..63))
3585          mediumConsumed(171),             -- Integer32 (-2..2147483647)
3586                                           -- AND
3587                                           -- JmJobStringTC (SIZE(0..63))
3588          colorantRequested(172),          -- Integer32 (-2..2147483647)
3589                                           -- AND/OR
3590                                           -- JmJobStringTC (SIZE(0..63))
3591          colorantConsumed(173),           -- Integer32 (-2..2147483647)
3592                                           -- AND/OR
3593                                           -- JmJobStringTC (SIZE(0..63))
3594          mediumTypeConsumed(174),         -- Integer32 (-2..2147483647)
3595                                           -- AND
3596                                           -- JmJobStringTC (SIZE(0..63))
3597          mediumSizeConsumed(175),         -- Integer32 (-2..2147483647)
3598                                           -- AND
3599                                           -- JmJobStringTC (SIZE(0..63))
3600
```

```
3601          -- Time attributes:
3602          jobSubmissionToServerTime(190), -- JmTimeStampTC
3603                                          -- AND/OR
3604                                          -- DateAndTime
3605          jobSubmissionTime(191),         -- JmTimeStampTC
3606                                          -- AND/OR
3607                                          -- DateAndTime
3608          jobStartedBeingHeldTime(192),   -- JmTimeStampTC
3609                                          -- AND/OR
3610                                          -- DateAndTime
3611          jobStartedProcessingTime(193),  -- JmTimeStampTC
3612                                          -- AND/OR
3613                                          -- DateAndTime
3614          jobCompletionTime(194),         -- JmTimeStampTC
3615                                          -- AND/OR
3616                                          -- DateAndTime
3617          jobProcessingCPUTime(195)       -- Integer32 (-2..2147483647)
3618      }
```

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

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

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

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

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

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

        print                                   0x4
            The job contains some instructions that specify printing

        scan                                    0x8
            The job contains some instructions that specify scanning
```

```
3666            faxIn                                   0x10
3667                The job contains some instructions that specify receive fax
3668
3669            faxOut                                  0x20
3670                The job contains some instructions that specify sending fax
3671
3672            getFile                                 0x40
3673                The job contains some instructions that specify accessing
3674                files or documents
3675
3676            putFile                                 0x80
3677                The job contains some instructions that specify storing
3678                files or documents
3679
3680            mailList                                0x100
3681                The job contains some instructions that specify
3682                distribution of documents using an electronic mail system.
3683
3684            These bit definitions are the equivalent of a type 2 enum
3685            except that combinations of them MAY be used together.  See
3686            section 3.7.1.2."
3687        SYNTAX      INTEGER (0..2147483647)   -- 31 bits, all but sign bit
3688
3689
3690
3691    JmJobStateReasons1TC ::= TEXTUAL-CONVENTION
3692        STATUS      current
3693        DESCRIPTION
3694            "The JmJobStateReasonsNTC (N=1..4) textual-conventions are used
3695            with the jmJobStateReasons1 object and jobStateReasonsN
3696            (N=2..4), respectively, to provide additional information
3697            regarding the current jmJobState object value.  These values
3698            MAY be used with any job state or states for which the reason
3699            makes sense.  See section 3.3.9.1 for the specification of each
3700            bit value defined for use with the JmJobStateReasons1TC.
3701
3702            NOTE - While values cannot be added to the jmJobState object
3703            without impacting deployed clients that take actions upon
3704            receiving jmJobState values, it is the intent that additional
3705            JmJobStateReasonsNTC enums can be defined and registered
3706            without impacting such deployed clients.  In other words, the
3707            jmJobStateReasons1 object and jobStateReasonsN attributes are
3708            intended to be extensible.
3709
3710            NOTE - The Job Monitoring MIB contains a superset of the IPP
3711            values[ipp-model] for the IPP 'job-state-reasons' attribute,
3712            since the Job Monitoring MIB is intended to cover other job
3713            submission protocols as well.  Also some of the names of the
3714            reasons have been changed from 'printer' to 'device', since the
3715            Job Monitoring MIB is intended to cover additional types of
3716            devices, including input devices, such as scanners.
3717
```

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

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

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

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

submissionInterrupted 0x8
The job was not completely submitted for some unforeseen reason, such as: (1) the server has crashed before the job was closed by the client, (2) the server or the document transfer method has crashed in some non-recoverable way before the document data was entirely transferred to the server, (3) the client crashed or failed to close the job before the time-out period.

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

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

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

jobProcessAfterSpecified 0x80
The value of the job's jobProcessAfterDateAndTime(51) attribute specifies a time that is still in the future. The job SHALL NOT be a candidate for processing until this

reason is removed and there are no other reasons to hold the job.

resourcesAreNotReady 0x100
At least one of the resources needed by the job, such as media, fonts, resource objects, etc., is not ready on any of the physical devices for which the job is a candidate. This condition MAY be detected when the job is accepted, or subsequently while the job is pending or processing, depending on implementation.

deviceStoppedPartly 0x200
One or more, but not all, of the devices to which the job is assigned are stopped. If all of the devices are stopped (or the only device is stopped), the deviceStopped reason SHALL be used.

deviceStopped 0x400
The device(s) to which the job is assigned is (are all) stopped.

jobInterpreting 0x800
The device to which the job is assigned is interpreting the document data.

jobPrinting 0x1000
The output device to which the job is assigned is marking media. This value is useful for servers and output devices which spend a great deal of time processing (1) when no marking is happening and then want to show that marking is now happening or (2) when the job is in the process of being canceled or aborted while the job remains in the processing state, but the marking has not yet stopped so that impression or sheet counts are still increasing for the job.

jobCanceledByUser 0x2000
The job was canceled by the owner of the job, i.e., by a user whose name is the same as the value of the job's jmJobOwner object, or by some other authorized end-user, such as a member of the job owner's security group.

jobCanceledByOperator 0x4000
The job was canceled by the operator, i.e., by a user who has been authenticated as having operator privileges (whether local or remote).

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

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

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

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

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

~~jobCompletedSuccessfully 0x80000~~
~~The job completed successfully.~~

~~jobCompletedWithWarnings 0x100000~~
~~The job completed with warnings.~~

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

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

~~jobPaused 0x400000~~
~~The job has been indefinitely suspended by a client issuing an operation to suspend the job so that other jobs may proceed using the same devices. The client MAY issue an operation to resume the paused job at any time, in which case the agent SHALL remove the jobPaused values from the job's jmJobStateReasons1 object and the job is eventually resumed at or near the point where the job was paused.~~

~~jobInterrupted 0x800000~~
~~The job has been interrupted while processing by a client issuing an operation that specifies another job to be run instead of the current job. The server or device will automatically resume the interrupted job when the interrupting job completes.~~

~~jobRetained 0x1000000~~
~~The job is being retained by the server or device with all of the job's document data (and submitted resources, such as fonts, logos, and forms, if any). Thus a client could issue an operation to the server or device to either (1) re-do the job (or a copy of the job) on the same server or device or (2) resubmit the job to another server or device. When a client could no longer re-do/resubmit the job, such as after the document data has been discarded, the agent SHALL remove the jobRetained value from the jmJobStateReasons1 object.~~

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

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

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

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

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

~~deletedByAdministrator 0x2~~
~~The administrator has deleted the job.~~

~~discardTimeArrived 0x4~~
~~The job has been deleted due to the fact that the time~~

~~specified by the job's job-discard-time attribute has arrived.~~

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

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

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

~~devicesNeedAttentionTimeOut 0x40~~
~~One or more document transforms that the job is using needs human intervention in order for the job to make progress, but the human intervention did not occur within the site-settable time-out value.~~

~~needsKeyOperatorTimeOut 0x80~~
~~One or more devices or document transforms that the job is using need a specially trained operator (who may need a key to unlock the device and gain access) in order for the job to make progress, but the key operator intervention did not occur within the site-settable time-out value.~~

~~jobStartWaitTimeOut 0x100~~
~~The server/device has stopped the job at the beginning of processing to await human action, such as installing a special cartridge or special non-standard media, but the job was not resumed within the site-settable time-out value and the server/device has transitioned the job to the pendingHeld state.~~

~~jobEndWaitTimeOut 0x200~~
~~The server/device has stopped the job at the end of processing to await human action, such as removing a special cartridge or restoring standard media, but the job was not resumed within the site-settable time-out value and the server/device has transitioned the job to the completed state.~~

~~jobPasswordWaitTimeOut 0x400~~
~~The server/device has stopped the job at the beginning of processing to await input of the job's password, but the~~

password was not received within the site-settable time-out value.

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

connectingToDeviceTimeOut 0x1000
The server is attempting to connect to one or more devices which may be dial-up, polled, or queued, and so may be busy with traffic from other systems, but server was unable to connect to the device within the site-settable time-out value.

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

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

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

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

jobPasswordWait 0x20000
The server/device has selected the job to be next to process, but instead of assigning resources and starting the job processing, the server/device has transitioned the job to the pendingHeld state to await entry of a password (and dispatched another job, if there is one).

validating 0x40000
The server/device is validating the job *after* accepting the job.

queueHeld 0x80000
The operator has held the entire job set or queue.

jobProofWait 0x100000
The job has produced a single proof copy and is in the pendingHeld state waiting for the requester to issue an operation to release the job to print normally, obeying any job and document copy attributes that were originally submitted.

~~heldForDiagnostics 0x200000~~
~~The system is running intrusive diagnostics, so that all jobs are being held.~~
~~noSpaceOnServer 0x800000~~
~~There is no room on the server to store all of the job.~~

~~pinRequired 0x1000000~~
~~The System Administrator settable device policy is (1) to require PINs, and (2) to hold jobs that do not have a pin supplied as an input parameter when the job was created.~~

~~exceededAccountLimit 0x2000000~~
~~The account for which this job is drawn has exceeded its limit. This condition SHOULD be detected before the job is scheduled so that the user does not wait until his/her job is scheduled only to find that the account is overdrawn. This condition MAY also occur while the job is processing either as processing begins or part way through processing.~~

~~heldForRetry 0x4000000~~
~~The job encountered some errors that the server/device could not recover from with its normal retry procedures, but the error might not be encountered if the job is processed again in the future. Example cases are phone number busy or remote file system in-accessible. For such a situation, the server/device SHALL transition the job from the processing to the pendingHeld, rather than to the aborted state.~~

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

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

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

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

~~badJob 0x40000000~~
~~This job was aborted by the system because this job has a major problem, such as an ill-formed PDL; the spooler SHOULD not even try another device.~~

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

```
4075            section 3.7.1.2. -- See the description under
4076            JmJobStateReasons1TC and the jobStateReasons2 attribute."
4077        SYNTAX      INTEGER (0..2147483647)   -- 31 bits, all but sign bit
4078
4079    JmJobStateReasons3TC ::= TEXTUAL-CONVENTION
4080        STATUS      current
4081        DESCRIPTION
4082            "This textual-convention is used with the jobStateReasons3
4083            attribute to provides additional information regarding the
4084            jmJobState object.  See section 3.3.9.3 for the specification
4085            of JmJobStateReasons3TC.  See section 3.3.9.1 for the
4086            description under JmJobStateReasons1TC for additional
4087            information that applies to all reasons.
4088
4089            The following standard values are defined (in hexadecimal) as
4090            powers of two, since multiple values may be used at the same
4091            time:
4092
4093            jobInterruptedByDeviceFailure      0x1
4094                A device or the print system software that the job was
4095                using has failed while the job was processing.  The server
4096                or device is keeping the job in the pendingHeld state until
4097                an operator can determine what to do with the job.
4098
4099            These bit definitions are the equivalent of a type 2 enum
4100            except that combinations of them may be used together.  See
4101            section 3.7.1.2.  The remaining bits are reserved for future
4102            standardization and/or registration.  See the description under
4103            JmJobStateReasons1TC and the jobStateReasons3 attribute."
4104        SYNTAX      INTEGER (0..2147483647)   -- 31 bits, all but sign bit
4105
4106
4107
4108
4109
4110    JmJobStateReasons4TC ::= TEXTUAL-CONVENTION
4111        STATUS      current
4112        DESCRIPTION
4113            "This textual-convention is used in the jobStateReasons4
4114            attribute to provides additional information regarding the
4115            jmJobState object.  See section 3.3.9.4 for the specification
4116            of JmJobStateReasons4TC.  See section 3.3.9.1 for the
4117            description under JmJobStateReasons1TC for additional
4118            information that applies to all reasons.
4119
4120            The following standard values are defined (in hexadecimal) as
4121            powers of two, since multiple values may be used at the same
4122            time:
4123
4124            none yet defined.  These bits are reserved for future
4125            standardization and/or registration.
4126
```

```
4127          These bit definitions are the equivalent of a type 2 enum
4128          except that combinations of them may be used together.  See
4129          section 3.7.1.2. -- See the description under
4130          JmJobStateReasons1TC and the jobStateReasons4 attribute."
4131      SYNTAX      INTEGER (0..2147483647)   -- 31 bits, all but sign bit
```

```
4132
4133
4134   jobmonMIBObjects  OBJECT IDENTIFIER  ::= { jobmonMIB 1 }
4135
4136   -- The General Group (MANDATORY)
4137
4138   -- The jmGeneralGroup consists entirely of the jmGeneralTable.
4139
4140   jmGeneral  OBJECT IDENTIFIER ::= { jobmonMIBObjects 1 }
4141
4142   jmGeneralTable  OBJECT-TYPE
4143       SYNTAX      SEQUENCE OF JmGeneralEntry
4144       MAX-ACCESS  not-accessible
4145       STATUS      current
4146       DESCRIPTION
4147           "The jmGeneralTable consists of information of a general nature
4148           that are per-job-set, but are not per-job.  See Section 2
4149           entitled 'Terminology and Job Model' for the definition of a
4150           job set.
4151
4152           The MANDATORY-GROUP macro specifies that this group is
4153           MANDATORY."
4154       ::= { jmGeneral 1 }
4155
4156
4157   jmGeneralEntry  OBJECT-TYPE
4158       SYNTAX      JmGeneralEntry
4159       MAX-ACCESS  not-accessible
4160       STATUS      current
4161       DESCRIPTION
4162           "Information about a job set (queue).
4163
4164           An entry SHALL exist in this table for each job set."
4165       INDEX  { jmGeneralJobSetIndex }
4166       ::= { jmGeneralTable 1 }
4167
4168
4169   JmGeneralEntry ::= SEQUENCE {
4170       jmGeneralJobSetIndex                Integer32 (1..32767),
4171       jmGeneralNumberOfActiveJobs         Integer32 (0..2147483647),
4172       jmGeneralOldestActiveJobIndex       Integer32 (0..2147483647),
4173       jmGeneralNewestActiveJobIndex       Integer32 (0..2147483647),
4174       jmGeneralJobPersistence             Integer32 (15..2147483647),
4175       jmGeneralAttributePersistence       Integer32 (15..2147483647),
4176       jmGeneralJobSetName                 JmUTF8StringTC (SIZE(0..63))
4177   }
```

```
4178
4179  jmGeneralJobSetIndex OBJECT-TYPE
4180      SYNTAX      Integer32 (1..32767)
4181      MAX-ACCESS  not-accessible
4182      STATUS      current
4183      DESCRIPTION
4184          "A unique value for each job set in this MIB.  The jmJobTable
4185          and jmAttributeTable tables have this same index as their
4186          primary index.
4187
4188          The value(s) of the jmGeneralJobSetIndex SHALL be persistent
4189          across power cycles, so that clients that have retained
4190          jmGeneralJobSetIndex values will access the same job sets upon
4191          subsequent power-up.
4192
4193          An implementation that has only one job set, such as a printer
4194          with a single queue, SHALL hard code this object with the value
4195          1.
4196
4197          See Section 2 entitled 'Terminology and Job Model' for the
4198          definition of a job set.
4199          Corresponds to the first index in jmJobTable and
4200          jmAttributeTable."
4201      ::= { jmGeneralEntry 1 }
4202
4203
4204  jmGeneralNumberOfActiveJobs OBJECT-TYPE
4205      SYNTAX      Integer32 (0..2147483647)
4206      MAX-ACCESS  read-only
4207      STATUS      current
4208      DESCRIPTION
4209          "The current number of 'active' jobs in the jmJobIDTable,
4210          jmJobTable, and jmAttributeTable, i.e., the total number of
4211          jobs that are in the pending, processing, or processingStopped
4212          states.  See the JmJobStateTC textual-convention for the exact
4213          specification of the semantics of the job states."
4214      DEFVAL      { 0 }      -- no jobs
4215      ::= { jmGeneralEntry 2 }
```

```
4216
4217   jmGeneralOldestActiveJobIndex  OBJECT-TYPE
4218       SYNTAX      Integer32 (0..2147483647)
4219       MAX-ACCESS  read-only
4220       STATUS      current
4221       DESCRIPTION
4222           "The jmJobIndex of the oldest job that is still in one of the
4223           'active' states (pending, processing, or processingStopped).
4224           In other words, the index of the 'active' job that has been in
4225           the job tables the longest.
4226
4227           If there are no active jobs, the agent SHALL set the value of
4228           this object to 0.
4229
4230           See Section 3.2 entitled 'The Job Tables and the Oldest Active
4231           and Newest Active Indexes' for a description of the usage of
4232           this object."
4233       DEFVAL      { 0 }      -- no active jobs
4234       ::= { jmGeneralEntry 3 }
4235
4236
4237
4238   jmGeneralNewestActiveJobIndex  OBJECT-TYPE
4239       SYNTAX      Integer32 (0..2147483647)
4240       MAX-ACCESS  read-only
4241       STATUS      current
4242       DESCRIPTION
4243           "The jmJobIndex of the newest job that is in one of the
4244           'active' states (pending, processing, or processingStopped).
4245           In other words, the index of the 'active' job that has been
4246           most recently added to the job tables.
4247
4248           When all jobs become 'inactive', i.e., enter the pendingHeld,
4249           completed, canceled, or aborted states, the agent SHALL set the
4250           value of this object to 0.
4251
4252           See Section 3.2 entitled 'The Job Tables and the Oldest Active
4253           and Newest Active Indexes' for a description of the usage of
4254           this object."
4255       DEFVAL      { 0 }      -- no active jobs
4256       ::= { jmGeneralEntry 4 }
```

```
4257
4258   jmGeneralJobPersistence OBJECT-TYPE
4259       SYNTAX      Integer32 (15..2147483647)
4260       UNITS       "seconds"
4261       MAX-ACCESS  read-only
4262       STATUS      current
4263       DESCRIPTION
4264           "The minimum time in seconds for this instance of the Job Set
4265           that an entry SHALL remain in the jmJobIDTable and jmJobTable
4266           after processing has completed, i.e., the minimum time in
4267           seconds starting when the job enters the completed, canceled,
4268           or aborted state.
4269
4270           Configuring this object is implementation-dependent.
4271
4272           This value SHALL be equal to or greater than the value of
4273           jmGeneralAttributePersistence.  This value SHOULD be at least
4274           60 which gives a monitoring or accounting application one
4275           minute in which to poll for job data."
4276       DEFVAL      { 60 }          -- one minute
4277       ::= { jmGeneralEntry 5 }
4278
4279
4280
4281   jmGeneralAttributePersistence OBJECT-TYPE
4282       SYNTAX      Integer32 (15..2147483647)
4283       UNITS       "seconds"
4284       MAX-ACCESS  read-only
4285       STATUS      current
4286       DESCRIPTION
4287           "The minimum time in seconds for this instance of the Job Set
4288           that an entry SHALL remain in the jmAttributeTable after
4289           processing has completed , i.e., the time in seconds starting
4290           when the job enters the completed, canceled, or aborted state.
4291
4292           Configuring this object is implementation-dependent.
4293
4294           This value SHOULD be at least 60 which gives a monitoring or
4295           accounting application one minute in which to poll for job
4296           data."
4297       DEFVAL      { 60 }          -- one minute
4298       ::= { jmGeneralEntry 6 }
```

```
4299
4300   jmGeneralJobSetName OBJECT-TYPE
4301       SYNTAX      JmUTF8StringTC (SIZE(0..63))
4302       MAX-ACCESS  read-only
4303       STATUS      current
4304       DESCRIPTION
4305           "The human readable name of this job set assigned by the system
4306           administrator (by means outside of this MIB).  Typically, this
4307           name SHOULD be the name of the job queue.  If a server or
4308           device has only a single job set, this object can be the
4309           administratively assigned name of the server or device itself.
4310           This name does not need to be unique, though each job set in a
4311           single Job Monitoring MIB SHOULD have distinct names.
4312
4313           NOTE - If the job set corresponds to a single printer and the
4314           Printer MIB is implemented, this value SHOULD be the same as
4315           the prtGeneralPrinterName object in the draft Printer MIB
4316           [print-mib-draft].  If the job set corresponds to an IPP
4317           Printer, this value SHOULD be the same as the IPP 'printer-
4318           name' Printer attribute.
4319
4320           NOTE - The purpose of this object is to help the user of the
4321           job monitoring application distinguish between several job sets
4322           in implementations that support more than one job set.
4323
4324           See the OBJECT compliance macro for the minimum maximum length
4325           required for conformance."
4326       DEFVAL      { ''H }      -- empty string
4327       ::= { jmGeneralEntry 7 }
4328
4329
4330
```

```
4331
4332
4333   -- The Job ID Group (MANDATORY)
4334
4335   -- The jmJobIDGroup consists entirely of the jmJobIDTable.
4336
4337   jmJobID  OBJECT IDENTIFIER ::= { jobmonMIBObjects 2 }
4338
4339   jmJobIDTable  OBJECT-TYPE
4340       SYNTAX      SEQUENCE OF JmJobIDEntry
4341       MAX-ACCESS  not-accessible
4342       STATUS      current
4343       DESCRIPTION
4344           "The jmJobIDTable provides a correspondence map (1) between the
4345           job submission ID that a client uses to refer to a job and (2)
4346           the jmGeneralJobSetIndex and jmJobIndex that the Job Monitoring
4347           MIB agent assigned to the job and that are used to access the
4348           job in all of the other tables in the MIB.  If a monitoring
4349           application already knows the jmGeneralJobSetIndex and the
4350           jmJobIndex of the job it is querying, that application NEED NOT
4351           use the jmJobIDTable.
4352
4353           The MANDATORY-GROUP macro specifies that this group is
4354           MANDATORY."
4355       ::= { jmJobID 1 }
4356
4357
4358
4359   jmJobIDEntry  OBJECT-TYPE
4360       SYNTAX      JmJobIDEntry
4361       MAX-ACCESS  not-accessible
4362       STATUS      current
4363       DESCRIPTION
4364           "The map from (1) the jmJobSubmissionID to (2) the
4365           jmGeneralJobSetIndex and jmJobIndex.
4366
4367           An entry SHALL exist in this table for each job currently known
4368           to the agent for all job sets and job states.  There MAY be
4369           more than one jmJobIDEntry that maps to a single job.  This
4370           many to one mapping can occur when more than one network entity
4371           along the job submission path supplies a job submission ID.
4372           See Section 3.5.  However, each job SHALL appear once and in
4373           one and only one job set."
4374       INDEX  { jmJobSubmissionID }
4375       ::= { jmJobIDTable 1 }
4376
4377   JmJobIDEntry ::= SEQUENCE {
4378       jmJobSubmissionID                   OCTET STRING(SIZE(48)),
4379       jmJobIDJobSetIndex                  Integer32 (0..32767),
4380       jmJobIDJobIndex                     Integer32 (0..2147483647)
4381   }
```

```
4382
4383  jmJobSubmissionID OBJECT-TYPE
4384      SYNTAX      OCTET STRING(SIZE(48))
4385      MAX-ACCESS  not-accessible
4386      STATUS      current
4387      DESCRIPTION
4388          "A quasi-unique 48-octet fixed-length string ID which
4389          identifies the job within a particular client-server
4390          environment.  There are multiple formats for the
4391          jmJobSubmissionID.  Each format SHALL be uniquely identified.
4392          See the JmJobSubmissionIDTypeTC textual convention.  Each
4393          format SHALL be registered using the procedures of a type 2
4394          enum.  See section 3.7.3 entitled: 'PWG Registration of Job
4395          Submission Id Formats'.
4396
4397          If the requester (client or server) does not supply a job
4398          submission ID in the job submission protocol, then the
4399          recipient (server or device) SHALL assign a job submission ID
4400          using any of the standard formats that have been reserved for
4401          agents and adding the final 8 octets to distinguish the ID from
4402          others submitted from the same requester.
4403
4404          The monitoring application, whether in the client or running
4405          separately, MAY use the job submission ID to help identify
4406          which jmJobIndex was assigned by the agent, i.e., in which row
4407          the job information is in the other tables.
4408
4409          NOTE - fixed-length is used so that a management application
4410          can use a shortened GetNext varbind (in SNMPv1 and SNMPv2) in
4411          order to get the next submission ID, disregarding the remainder
4412          of the ID in order to access jobs independent of the trailing
4413          identifier part, e.g., to get all jobs submitted by a
4414          particular jmJobOwner or submitted from a particular MAC
4415          address.
4416
4417          See the JmJobSubmissionIDTypeTC textual convention.
4418          See APPENDIX B - Support of Job Submission Protocols."
4419      ::= { jmJobIDEntry 1 }
```

```
4420
4421   jmJobIDJobSetIndex OBJECT-TYPE
4422       SYNTAX      Integer32 (0..32767)
4423       MAX-ACCESS  read-only
4424       STATUS      current
4425       DESCRIPTION
4426           "This object contains the value of the jmGeneralJobSetIndex for
4427           the job with the jmJobSubmissionID value, i.e., the job set
4428           index of the job set in which the job was placed when that
4429           server or device accepted the job.  This 16-bit value in
4430           combination with the jmJobIDJobIndex value permits the
4431           management application to access the other tables to obtain the
4432           job-specific objects for this job.
4433
4434           See jmGeneralJobSetIndex in the jmGeneralTable."
4435       DEFVAL      { 0 }      -- 0 indicates no job set index
4436       ::= { jmJobIDEntry 2 }
4437
4438
4439
4440   jmJobIDJobIndex OBJECT-TYPE
4441       SYNTAX      Integer32 (0..2147483647)
4442       MAX-ACCESS  read-only
4443       STATUS      current
4444       DESCRIPTION
4445           "This object contains the value of the jmJobIndex for the job
4446           with the jmJobSubmissionID value, i.e., the job index for the
4447           job when the server or device accepted the job.  This value, in
4448           combination with the jmJobIDJobSetIndex value, permits the
4449           management application to access the other tables to obtain the
4450           job-specific objects for this job.
4451
4452           See jmJobIndex in the jmJobTable."
4453       DEFVAL      { 0 }      -- 0 indicates no jmJobIndex value.
4454       ::= { jmJobIDEntry 3 }
4455
4456
```

```
4457
4458
4459   -- The Job Group (MANDATORY)
4460
4461   -- The jmJobGroup consists entirely of the jmJobTable.
4462
4463   jmJob  OBJECT IDENTIFIER ::= { jobmonMIBObjects 3 }
4464
4465   jmJobTable  OBJECT-TYPE
4466       SYNTAX      SEQUENCE OF JmJobEntry
4467       MAX-ACCESS  not-accessible
4468       STATUS      current
4469       DESCRIPTION
4470           "The jmJobTable consists of basic job state and status
4471           information for each job in a job set that (1) monitoring
4472           applications need to be able to access in a single SNMP Get
4473           operation, (2) that have a single value per job, and (3) that
4474           SHALL always be implemented.
4475
4476           The MANDATORY-GROUP macro specifies that this group is
4477           MANDATORY."
4478       ::= { jmJob 1 }
4479
4480
4481
4482   jmJobEntry  OBJECT-TYPE
4483       SYNTAX      JmJobEntry
4484       MAX-ACCESS  not-accessible
4485       STATUS      current
4486       DESCRIPTION
4487           "Basic per-job state and status information.
4488
4489           An entry SHALL exist in this table for each job, no matter what
4490           the state of the job is.  Each job SHALL appear in one and only
4491           one job set.
4492
4493           See Section 3.2 entitled 'The Job Tables'."
4494       INDEX  { jmGeneralJobSetIndex, jmJobIndex }
4495       ::= { jmJobTable 1 }
4496
4497   JmJobEntry ::= SEQUENCE {
4498       jmJobIndex                          Integer32 (1..2147483647),
4499       jmJobState                          JmJobStateTC,
4500       jmJobStateReasons1                  JmJobStateReasons1TC,
4501       jmNumberOfInterveningJobs           Integer32 (-2..2147483647),
4502       jmJobKOctetsPerCopyRequested        Integer32 (-2..2147483647),
4503       jmJobKOctetsProcessed               Integer32 (-2..2147483647),
4504       jmJobImpressionsPerCopyRequested    Integer32 (-2..2147483647),
4505       jmJobImpressionsCompleted           Integer32 (-2..2147483647),
4506       jmJobOwner                          JmJobStringTC (SIZE(0..63))
4507   }
```

```
4508
4509   jmJobIndex OBJECT-TYPE
4510       SYNTAX      Integer32 (1..2147483647)
4511       MAX-ACCESS  not-accessible
4512       STATUS      current
4513       DESCRIPTION
4514           "The sequential, monatonically increasing identifier index for
4515           the job generated by the server or device when that server or
4516           device accepted the job.  This index value permits the
4517           management application to access the other tables to obtain the
4518           job-specific row entries.
4519
4520           See Section 3.2 entitled 'The Job Tables and the Oldest Active
4521           and Newest Active Indexes'.
4522           See Section 3.5 entitled 'Job Identification'.
4523           See also jmGeneralNewestActiveJobIndex for the largest value of
4524           jmJobIndex.
4525           See JmJobSubmissionIDTypeTC for a limit on the size of this
4526           index if the agent represents it as an 8-digit decimal number."
4527       ::= { jmJobEntry 1 }
4528
4529
4530
4531   jmJobState OBJECT-TYPE
4532       SYNTAX      JmJobStateTC
4533       MAX-ACCESS  read-only
4534       STATUS      current
4535       DESCRIPTION
4536           "The current state of the job (pending, processing, completed,
4537           etc.).  Agents SHALL implement only those states which are
4538           appropriate for the particular implementation.  However,
4539           management applications SHALL be prepared to receive all the
4540           standard job states.
4541
4542           The final value for this object SHALL be one of: completed,
4543           canceled, or aborted.  The minimum length of time that the
4544           agent SHALL maintain MIB data for a job in the completed,
4545           canceled, or aborted state before removing the job data from
4546           the jmJobIDTable and jmJobTable is specified by the value of
4547           the jmGeneralJobPersistence object."
4548       DEFVAL      { unknown }      -- default is unknown
4549       ::= { jmJobEntry 2 }
```

```
4550
4551   jmJobStateReasons1 OBJECT-TYPE
4552       SYNTAX      JmJobStateReasons1TC
4553       MAX-ACCESS  read-only
4554       STATUS      current
4555       DESCRIPTION
4556           "Additional information about the job's current state, i.e.,
4557           information that augments the value of the job's jmJobState
4558           object.
4559
4560           Implementation of any reason values is OPTIONAL, but an agent
4561           SHOULD return any reason information available.  These values
4562           MAY be used with any job state or states for which the reason
4563           makes sense.  Since the Job State Reasons will be more dynamic
4564           than the Job State, it is recommended that a job monitoring
4565           application read this object every time jmJobState is read.
4566           When the agent cannot provide a reason for the current state of
4567           the job, the value of the jmJobStateReasons1 object and
4568           jobStateReasonsN attributes SHALL be 0.
4569
4570           The jobStateReasonsN (N=2..4) attributes provide further
4571           additional information about the job's current state."
4572       DEFVAL      { 0 }      -- no reasons
4573       ::= { jmJobEntry 3 }
4574
4575
4576
4577   jmNumberOfInterveningJobs OBJECT-TYPE
4578       SYNTAX      Integer32 (-2..2147483647)
4579       MAX-ACCESS  read-only
4580       STATUS      current
4581       DESCRIPTION
4582           "The number of jobs that are expected to complete processing
4583           before this job has completed processing according to the
4584           implementation's queuing algorithm, if no other jobs were to be
4585           submitted.  In other words, this value is the job's queue
4586           position.  The agent SHALL return a value of 0 for this
4587           attribute when the job is the next job to complete processing
4588           (or has completed processing)."
4589       DEFVAL      { 0 }      -- default is no intervening jobs.
4590       ::= { jmJobEntry 4 }
```

```
4591
4592   jmJobKOctetsPerCopyRequested OBJECT-TYPE
4593       SYNTAX      Integer32 (-2..2147483647)
4594       MAX-ACCESS  read-only
4595       STATUS      current
4596       DESCRIPTION
4597           "The total size in K (1024) octets of the document(s) being
4598           requested to be processed in the job.  The agent SHALL round
4599           the actual number of octets up to the next highest K.  Thus 0
4600           octets is represented as '0', 1-1024 octets is represented as
4601           '1', 1025-2048 is represented as '2', etc.
4602
4603           In computing this value, the server/device SHALL NOT include
4604           the multiplicative factors contributed by (1) the number of
4605           document copies, and (2) the number of job copies, independent
4606           of whether the device can process multiple copies of the job or
4607           document without making multiple passes over the job or
4608           document data and independent of whether the output is collated
4609           or not.  Thus the server/device computation is independent of
4610           the implementation and indicates the size of the document(s)
4611           measured in K octets independent of the number of copies."
4612       DEFVAL      { -2 }      -- the default is unknown(-2)
4613       ::= { jmJobEntry 5 }
4614
4615
4616
4617   jmJobKOctetsProcessed OBJECT-TYPE
4618       SYNTAX      Integer32 (-2..2147483647)
4619       MAX-ACCESS  read-only
4620       STATUS      current
4621       DESCRIPTION
4622           "The total number of octets processed by the server or device
4623           measured in units of K (1024) octets so far.  The agent SHALL
4624           round the actual number of octets processed up to the next
4625           higher K.  Thus 0 octets is represented as '0', 1-1024 octets
4626           is represented as '1', 1025-2048 octets is '2', etc.  For
4627           printing devices, this value is the number interpreted by the
4628           page description language interpreter rather than what has been
4629           marked on media.
4630
4631           For implementations where multiple copies are produced by the
4632           interpreter with only a single pass over the data, the final
4633           value SHALL be equal to the value of the
4634           jmJobKOctetsPerCopyRequested object.  For implementations where
4635           multiple copies are produced by the interpreter by processing
4636           the data for each copy, the final value SHALL be a multiple of
4637           the value of the jmJobKOctetsPerCopyRequested object.
4638
4639           NOTE - See the impressionsCompletedCurrentCopy and
4640           pagesCompletedCurrentCopy attributes for attributes that are
4641           reset on each document copy.
4642
```

```
4643            NOTE - The jmJobKOctetsProcessed object can be used with the
4644            jmJobKOctetsPerCopyRequested object to provide an indication of
4645            the relative progress of the job, provided that the
4646            multiplicative factor is taken into account for some
4647            implementations of multiple copies."
4648        DEFVAL      { 0 }      -- default is no octets processed.
4649        ::= { jmJobEntry 6 }
4650
4651
4652    jmJobImpressionsPerCopyRequested OBJECT-TYPE
4653        SYNTAX      Integer32 (-2..2147483647)
4654        MAX-ACCESS  read-only
4655        STATUS      current
4656        DESCRIPTION
4657            "The total size in number of impressions of the document(s)
4658            submitted.
4659
4660            In computing this value, the server/device SHALL NOT include
4661            the multiplicative factors contributed by (1) the number of
4662            document copies, and (2) the number of job copies, independent
4663            of whether the device can process multiple copies of the job or
4664            document without making multiple passes over the job or
4665            document data and independent of whether the output is collated
4666            or not.  Thus the server/device computation is independent of
4667            the implementation and reflects the size of the document(s)
4668            measured in impressions independent of the number of copies.
4669
4670            See the definition of the term 'impression' in Section 2."
4671        DEFVAL      { -2 }      -- default is unknown(-2)
4672        ::= { jmJobEntry 7 }
4673
4674
4675    jmJobImpressionsCompleted OBJECT-TYPE
4676        SYNTAX      Integer32 (-2..2147483647)
4677        MAX-ACCESS  read-only
4678        STATUS      current
4679        DESCRIPTION
4680            "The total number of impressions completed for this job so far.
4681            For printing devices, the impressions completed includes
4682            interpreting, marking, and stacking the output.  For other
4683            types of job services, the number of impressions completed
4684            includes the number of impressions processed.
4685
4686            NOTE - See the impressionsCompletedCurrentCopy and
4687            pagesCompletedCurrentCopy attributes for attributes that are
4688            reset on each document copy.
4689
4690            NOTE - The jmJobImpressionsCompleted object can be used with
4691            the jmJobImpressionsPerCopyRequested object to provide an
4692            indication of the relative progress of the job, provided that
4693            the multiplicative factor is taken into account for some
4694            implementations of multiple copies.
```

```
4695
4696            See the definition of the term 'impression' in Section 2 and
4697            the counting example in Section 3.4 entitled 'Monitoring Job
4698            Progress'."
4699        DEFVAL      { 0 }      -- default is no octets
4700        ::= { jmJobEntry 8 }
4701
4702
4703
4704    jmJobOwner OBJECT-TYPE
4705        SYNTAX      JmJobStringTC (SIZE(0..63))
4706        MAX-ACCESS  read-only
4707        STATUS      current
4708        DESCRIPTION
4709            "The coded character set name of the user that submitted the
4710            job.  The method of assigning this user name will be system
4711            and/or site specific but the method MUST ensure that the name
4712            is unique to the network that is visible to the client and
4713            target device.
4714
4715            This value SHOULD be the most authenticated name of the user
4716            submitting the job.
4717
4718            See the OBJECT compliance macro for the minimum maximum length
4719            required for conformance."
4720        DEFVAL      { ''H }      -- default is empty string
4721        ::= { jmJobEntry 9 }
4722
4723
```

```
4724
4725
4726   -- The Attribute Group (MANDATORY)
4727
4728   -- The jmAttributeGroup consists entirely of the jmAttributeTable.
4729   --
4730   -- Implementation of the objects in this group is MANDATORY.
4731   -- See Section 3.1 entitled 'Conformance Considerations'.
4732   -- An agent SHALL implement any attribute if (1) the server or device
4733   -- supports the functionality represented by the attribute and (2) the
4734   -- information is available to the agent.
4735
4736   jmAttribute  OBJECT IDENTIFIER ::= { jobmonMIBObjects 4 }
4737
4738
4739
4740   jmAttributeTable  OBJECT-TYPE
4741       SYNTAX      SEQUENCE OF JmAttributeEntry
4742       MAX-ACCESS  not-accessible
4743       STATUS      current
4744       DESCRIPTION
4745           "The jmAttributeTable SHALL contain attributes of the job and
4746           document(s) for each job in a job set.  Instead of allocating
4747           distinct objects for each attribute, each attribute is
4748           represented as a separate row in the jmAttributeTable.
4749
4750           The MANDATORY-GROUP macro specifies that this group is
4751           MANDATORY.  An agent SHALL implement any attribute if (1) the
4752           server or device supports the functionality represented by the
4753           attribute and (2) the information is available to the agent. "
4754       ::= { jmAttribute 1 }
4755
4756
4757
```

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

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

        The agent SHALL create rows in the jmAttributeTable as the
        server or device is able to discover the attributes either from
        the job submission protocol itself or from the document PDL.
        As the documents are interpreted, the interpreter MAY discover
        additional attributes and so the agent adds additional rows to
        this table.  As the attributes that represent resources are
        actually consumed, the usage counter contained in the
        jmAttributeValueAsInteger object is incremented according to
        the units indicated in the description of the JmAttributeTypeTC
        enum.

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

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

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

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

```
4808
4809    jmAttributeTypeIndex OBJECT-TYPE
4810        SYNTAX      JmAttributeTypeTC
4811        MAX-ACCESS  not-accessible
4812        STATUS      current
4813        DESCRIPTION
4814            "The type of attribute that this row entry represents.
4815
4816            The type MAY identify information about the job or document(s)
4817            or MAY identify a resource required to process the job before
4818            the job start processing and/or consumed by the job as the job
4819            is processed.
4820
4821            Examples of job attributes (i.e., apply to the job as a whole)
4822            that have only one instance per job include:
4823            jobCopiesRequested(90), documentCopiesRequested(92),
4824            jobCopiesCompleted(91), documentCopiesCompleted(93), while
4825            examples of job attributes that may have more than one instance
4826            per job include:  documentFormatIndex(37), and
4827            documentFormat(38).
4828
4829            Examples of document attributes (one instance per document)
4830            include: fileName(34), and documentName(35).
4831
4832            Examples of required and consumed resource attributes include:
4833            pagesRequested(130), mediumRequested(170), pagesCompleted(131),
4834            and mediumConsumed(171), respectively."
4835        ::= { jmAttributeEntry 1 }
4836
4837
4838
4839    jmAttributeInstanceIndex OBJECT-TYPE
4840        SYNTAX      Integer32 (1..32767)
4841        MAX-ACCESS  not-accessible
4842        STATUS      current
4843        DESCRIPTION
4844            "A running 16-bit index of the attributes of the same type for
4845            each job.  For those attributes with only a single instance per
4846            job, this index value SHALL be 1.  For those attributes that
4847            are a single value per document, the index value SHALL be the
4848            document number, starting with 1 for the first document in the
4849            job.  Jobs with only a single document SHALL use the index
4850            value of 1.  For those attributes that can have multiple values
4851            per job or per document, such as documentFormatIndex(37) or
4852            documentFormat(38), the index SHALL be a running index for the
4853            job as a whole, starting at 1."
4854        ::= { jmAttributeEntry 2 }
```

```
4855
4856   jmAttributeValueAsInteger OBJECT-TYPE
4857       SYNTAX      Integer32 (-2..2147483647)
4858       MAX-ACCESS  read-only
4859       STATUS      current
4860       DESCRIPTION
4861           "The integer value of the attribute.  The value of the
4862           attribute SHALL be represented as an integer if the enum
4863           description in the JmAttributeTypeTC textual-convention
4864           definition has the tag: 'INTEGER:'.
4865
4866           Depending on the enum definition, this object value MAY be an
4867           integer, a counter, an index, or an enum, depending on the
4868           jmAttributeTypeIndex value.  The units of this value are
4869           specified in the enum description.
4870
4871           For those attributes that are accumulating job consumption as
4872           the job is processed as specified in the JmAttributeTypeTC
4873           textual-convention, SHALL contain the final value after the job
4874           completes processing, i.e., this value SHALL indicate the total
4875           usage of this resource made by the job.
4876
4877           A monitoring application is able to copy this value to a
4878           suitable longer term storage for later processing as part of an
4879           accounting system.
4880
4881           Since the agent MAY add attributes representing resources to
4882           this table while the job is waiting to be processed or being
4883           processed, which can be a long time before any of the resources
4884           are actually used, the agent SHALL set the value of the
4885           jmAttributeValueAsInteger object to 0 for resources that the
4886           job has not yet consumed.
4887
4888           Attributes for which the concept of an integer value is
4889           meaningless, such as fileName(34), jobName, and
4890           processingMessage, do not have the 'INTEGER:' tag in the
4891           JmAttributeTypeTC definition and so an agent SHALL always
4892           return a value of '-1' to indicate 'other' for the value of the
4893           jmAttributeValueAsInteger object for these attributes.
4894
4895           For attributes which do have the 'INTEGER:' tag in the
4896           JmAttributeTypeTC definition, if the integer value is not (yet)
4897           known, the agent either (1) SHALL not materialize the row in
4898           the jmAttributeTable until the value is known or (2) SHALL
4899           return a '-2' to represent an 'unknown' counting integer value,
4900           a '0' to represent an 'unknown' index value, and a '2' to
4901           represent an 'unknown(2)' enum value."
4902       DEFVAL      { -2 }      -- default value is unknown(-2)
4903       ::= { jmAttributeEntry 3 }
```

```
4904
4905   jmAttributeValueAsOctets OBJECT-TYPE
4906       SYNTAX      OCTET STRING(SIZE(0..63))
4907       MAX-ACCESS  read-only
4908       STATUS      current
4909       DESCRIPTION
4910           "The octet string value of the attribute.  The value of the
4911           attribute SHALL be represented as an OCTET STRING if the enum
4912           description in the JmAttributeTypeTC textual-convention
4913           definition has the tag: 'OCTETS:'.
4914
4915           Depending on the enum definition, this object value MAY be a
4916           coded character set string (text), such as 'JmUTF8StringTC', or
4917           a binary octet string, such as 'DateAndTime'.
4918
4919           Attributes for which the concept of an octet string value is
4920           meaningless, such as pagesCompleted, do not have the tag
4921           'OCTETS:' in the JmAttributeTypeTC definition and so the agent
4922           SHALL always return a zero length string for the value of the
4923           jmAttributeValueAsOctets object.
4924
4925           For attributes which do have the 'OCTETS:' tag in the
4926           JmAttributeTypeTC definition, if the OCTET STRING value is not
4927           (yet) known, the agent either SHALL NOT materialize the row in
4928           the jmAttributeTable until the value is known or SHALL return a
4929           zero-length string."
4930       DEFVAL      { ''H }      -- empty string
4931       ::= { jmAttributeEntry 4 }
```

```
4932
4933
4934    -- The Mirror Attribute Group (OPTIONAL)
4935
4936    -- The jmMirrorAttrGroup consists entirely of the jmMirrorAttrTable.
4937    --
4938    -- Implementation of the objects in this group is OPTIONAL.
4939    -- See Section 3.1 entitled 'Conformance Considerations'.
4940    -- The jmMirrorAttrTable complements the MANDATORY jmAttributeTable.
4941    --
4942    -- The jmMirrorAttrTable provides access to all of the attributes that
4943    -- an implementation supports, sorted by attribute type (traditional
4944    -- SNMP MIB access), rather than being sorted by job set and job index
4945    -- (modern object-oriented access) as in the analogous
4946    -- jmAttributeTable.
4947
4948    jmMirrorAttr    OBJECT IDENTIFIER ::= { jobmonMIBObjects 5 }
4949
4950    jmMirrorAttrTable  OBJECT-TYPE
4951        SYNTAX      SEQUENCE OF JmMirrorAttrEntry
4952        MAX-ACCESS  not-accessible
4953        STATUS      current
4954        DESCRIPTION
4955            "The jmMirrorAttrTable is an OPTIONAL table which provides
4956            identical attributes to the jmAttributeTable but with a
4957            different index structure.  See jmAttributeTable for further
4958            details.
4959
4960            See Section 3.3 entitled 'The Attribute Mechanism' for a
4961            description of the jmMirrorAttrTable."
4962        ::= { jmMirrorAttr 1 }
4963
4964
4965
```

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

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

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

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

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

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

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

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

```
5016
5017    jmMirrorAttrValueAsInteger OBJECT-TYPE
5018        SYNTAX      Integer32 (-2..2147483647)
5019        MAX-ACCESS  read-only
5020        STATUS      current
5021        DESCRIPTION
5022            "The integer value of the attribute.
5023
5024            See jmAttributeValueAsInteger in jmAttributeTable for complete
5025            description."
5026        DEFVAL      { -2 }      -- default value is unknown(-2)
5027        ::= { jmMirrorAttrEntry 3 }
5028
5029    jmMirrorAttrValueAsOctets OBJECT-TYPE
5030        SYNTAX      OCTET STRING(SIZE(0..63))
5031        MAX-ACCESS  read-only
5032        STATUS      current
5033        DESCRIPTION
5034            "The octet string value of the attribute.
5035
5036            See jmAttributeValueAsOctets in jmAttributeTable for complete
5037            description."
5038        DEFVAL      { ''H }      -- empty string
5039        ::= { jmMirrorAttrEntry 4 }
```

```
5040
5041
5042    -- The System Group (MANDATORY)
5043    -- (This group was added in version 1.3 of this MIB).
5044
5045    -- The jmMirrorAttrGroup consists entirely of objects that summarize
5046    -- the implementation of this MIB on a system.
5047
5048    jmSystem        OBJECT IDENTIFIER ::= { jobmonMIBObjects 6 }
5049
5050    jmSystemVersionString OBJECT-TYPE
5051        SYNTAX      JmUTF8StringTC
5052        MAX-ACCESS  read-only
5053        STATUS      current
5054        DESCRIPTION
5055            "The minor and minor version of this MIB implemented by this
5056            system.
5057
5058            The format of the string SHALL be the ASCII major version
5059            number followed by an ASCII PERIOD (.), followed by the ASCII
5060            minor version number, i.e., '1.3' for this version."
5061        DEFVAL  { '312E33'H }                   -- version 1.3
5062        ::= { jmSystem 1 }
5063
5064    jmSystemOptionSupport OBJECT-TYPE
5065        SYNTAX      INTEGER(0..2147483647)    -- biggest int 2**31 - 1
5066        MAX-ACCESS  read-only
5067        STATUS      current
5068        DESCRIPTION
5069            "The options of the MIB specification that this implementation
5070            supports specified as a bit mask.
5071
5072            The current set of values (which may be extended in the future)
5073            is given below:
5074
5075                   1 : jmMirrorAttrGroup                -- 2**0   OPTIONAL
5076
5077            Example:  An implementation supporting the jmMirrorAttrGroup
5078            would return an integer value of { 1 }.
5079
5080            This object helps a management application determine which MIB
5081            options are supported in this system."
5082        DEFVAL    { 0 }                         -- no options are required
5083        ::= { jmSystem 2 }
5084
```

```
5085
5086    jmSystemAttrIntegerSupport OBJECT-TYPE
5087        SYNTAX      OCTET STRING (SIZE (0..63))
5088        MAX-ACCESS  read-only
5089        STATUS      current
5090        DESCRIPTION
5091            "A bit array indicating which attributes of the MIB this
5092            implementation supports with meaningful integer values.
5093
5094            The value of this object is a sparse bit array in which bit n
5095            is a 1 if attribute n is supported with the
5096            jmAttributeValueAsInteger object with meaningful values, where
5097            n is the value of the enumerated attribute type in the
5098            JmAttributeTypeTC used in jmAttributeTypeIndex (and the
5099            jmMirrorAttrTypeIndex if the jmMirrorAttrTable is implemented).
5100            Bit n MUST be 0 (or beyond the end of the returned bit array),
5101            if attribute n is not supported or is always returned with a '-
5102            1'(other) or '-2'(unknown) value.
5103
5104            The high order bit of the first octet in this octet string
5105            corresponds to an attribute type of 0 (reserved), i.e., the bit
5106            string uses the Big Endian numbering convention.  Compare with
5107            the BITS data type in SMIv2 [SMIv2-SMI] which has the same
5108            format but requires contiguous enumerated bits.  Trailing
5109            octets in the octet string that contain only zero bits MUST NOT
5110            be returned.
5111
5112            Note:  private attributes cannot be represented in this bit
5113            array because their enum values are in the range 2**30 to
5114            2**31-1.  See section 3.3.8.
5115
5116            Example:  An implementation supporting the attributes:
5117            jobStateReasons2(3), jobStateReasons3(4), and jobName(23)
5118            would return a one-octet string value of { '18'H }, since
5119            jobName is an octet string value, not an integer value.
5120
5121            This object helps a management application determine which
5122            attributes with meaningful integer values MAY be present on
5123            jobs in this system."
5124        DEFVAL   { ''H }                    -- no attributes are required
5125        ::= { jmSystem 3 }
5126
```

```
5127
5128  jmSystemAttrOctetsSupport OBJECT-TYPE
5129      SYNTAX      OCTET STRING (SIZE (0..63))
5130      MAX-ACCESS  read-only
5131      STATUS      current
5132      DESCRIPTION
5133          "A bit array indicating which attributes of the MIB this
5134          implementation supports with meaningful octet string values.
5135
5136          The format and semantics of this object is the same as
5137          jmSystemAttrIntegerSupport, except that bit n indicates that
5138          attribute n supports the jmAttributeValueAsOctets object with
5139          meaningful values, instead of the jmAttributeValueAsInteger
5140          object.  Bit n MUST be 0 (or beyond the end of the returned bit
5141          array), if attribute n is not supported or is always returned
5142          as a zero-length octet string value.
5143
5144          If an implementation supports both jmAttributeValueAsInteger
5145          and jmAttributeValueAsOctets with meaningful values for
5146          attribute n, bit n MUST appear in both bit array objects with a
5147          1 value.
5148
5149          Example:  An implementation supporting the attributes:
5150          jobStateReasons2(3), jobStateReasons3(4), and jobName(23)
5151          would return a three-octet string value of { '000001'H }, since
5152          jobStateReasons2 and jobStateReasons3 are integer values, not
5153          octet string values.
5154
5155          This object helps a management application determine which
5156          attributes with meaningful octet string values MAY be present
5157          on jobs in this system."
5158      DEFVAL   { ''H }                    -- no attributes are required
5159      ::= { jmSystem 4 }
5160
```

```
5161
5162    jmSystemAttrMultiRowSupport OBJECT-TYPE
5163        SYNTAX      OCTET STRING (SIZE (0..63))
5164        MAX-ACCESS  read-only
5165        STATUS      current
5166        DESCRIPTION
5167            "A bit array indicating which MULTI-ROW attributes of the MIB
5168            this implementation supports with multiple integer values
5169            and/or multiple octet string values.
5170
5171            The format of this object is the same as the
5172            jmSystemAttrIntegerSupport and jmSystemAttrOctetsSupport
5173            objects.  Bit n MUST be 1, if attribute n is actually supported
5174            with more than one integer and/or more than one octet string
5175            value.  Bit n MUST be 0 (or beyond the end of the returned bit
5176            array), if attribute n is not supported, is always returned as
5177            a single integer value, or as a single octet string value.  For
5178            every bit n that is a 1 in this bit array, there MUST be a
5179            corresponding 1 for bit n in either jmSystemAttrIntegerSupport,
5180            jmSystemAttrOctetsSupport, or both.
5181
5182            Example:  Consider an implementation supporting:
5183            (a) the jobStateReasons2(3), jobStateReasons3(4) SINGLE-ROW
5184            integer attributes
5185            (b) the jobName(23) SINGLE-ROW octet string attribute
5186            (c) more than one integer value for the mediumRequested(170)
5187            and mediumConsumed(171) MULTI-ROW attributes AND
5188            (d) more than one octet string value for the fileName(34),
5189            documentName(35), and mediumConsumed(171) MULTI-ROW attributes
5190            (e) no octet string values for mediumRequested(170).
5191            Such an implementation would return:
5192            jmSystemAttrIntegerSupport 22 octets:
5193              { '18000000 00000000 00000000 00000000 00000000 0030'H }
5194            jmSystemAttrOctetsSupport 22 octets:
5195              { '00000100 30000000 00000000 00000000 00000000 0010'H }
5196            jmSystemAttrMultiRowSupport 22 octets:
5197              { '00000000 30000000 00000000 00000000 00000000 0030'H }
5198
5199            Example:  Consider an implementation that supports the
5200            fileName(34) MULTI-ROW attribute, but does not support more
5201            than one document per job.  Such an implementation would NOT
5202            return a 1 bit for bit 34 in jmSystemAttrMultiRowSupport, since
5203            such an implementation would never return more than one
5204            fileName value for a job.  It would return a zero-length
5205            string, since it never returns more than one value.
5206
5207            This object helps a management application determine which
5208            attributes may return more than one integer value or more than
5209            one octet string value on jobs in this system."
5210        DEFVAL   { ''H }                    -- no attributes are required
5211        ::= { jmSystem 5 }
```

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

jobmonMIBNotifications  OBJECT IDENTIFIER  ::= { jobmonMIB 2 }



-- Conformance Information

jmMIBConformance OBJECT IDENTIFIER ::= { jobmonMIB 3 }



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

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

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

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

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

-- There are no CONDITIONALLY MANDATORY or OPTIONAL groups.

    ::= { jmMIBConformance 1 }
```

```
5261   jmMIBGroups      OBJECT IDENTIFIER ::= { jmMIBConformance 2 }
5262
5263   jmGeneralGroup OBJECT-GROUP
5264       OBJECTS {
5265           jmGeneralNumberOfActiveJobs,   jmGeneralOldestActiveJobIndex,
5266           jmGeneralNewestActiveJobIndex, jmGeneralJobPersistence,
5267           jmGeneralAttributePersistence, jmGeneralJobSetName}
5268       STATUS  current
5269       DESCRIPTION
5270           "The general group."
5271       ::= { jmMIBGroups 1 }
5272
5273
5274
5275   jmJobIDGroup OBJECT-GROUP
5276       OBJECTS {
5277           jmJobIDJobSetIndex, jmJobIDJobIndex }
5278       STATUS  current
5279       DESCRIPTION
5280           "The job ID group."
5281       ::= { jmMIBGroups 2 }
5282
5283
5284
5285   jmJobGroup OBJECT-GROUP
5286       OBJECTS {
5287           jmJobState, jmJobStateReasons1, jmNumberOfInterveningJobs,
5288           jmJobKOctetsPerCopyRequested, jmJobKOctetsProcessed,
5289           jmJobImpressionsPerCopyRequested, jmJobImpressionsCompleted,
5290           jmJobOwner }
5291       STATUS  current
5292       DESCRIPTION
5293           "The job group."
5294       ::= { jmMIBGroups 3 }
5295
5296
5297
5298   jmAttributeGroup OBJECT-GROUP
5299       OBJECTS {
5300           jmAttributeValueAsInteger, jmAttributeValueAsOctets }
5301       STATUS  current
5302       DESCRIPTION
5303           "The attribute group."
5304       ::= { jmMIBGroups 4 }
5305
5306
```

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

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


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


END
```

5 Appendix A - Implementing the Job Life Cycle

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

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

The job states are intended to be user visible. The agent SHALL make these states visible in the MIB, but only for the subset of job states that the implementation has. Some implementations MAY need to have sub-states of these user-visible states. The jmJobStateReasons1 object and the jobStateReasons*N* (*N*=2..4) attributes can be used to represent the sub-states of the jobs.

Job states are intended to last a user-visible length of time in most implementations. However, some jobs may pass through some states in zero time in some situations and/or in some implementations.

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

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

6 APPENDIX B - Support of Job Submission Protocols

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

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

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

7 References

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

~~[char-set-policy] Harald Avelstrand, "IETF Policy on Character Sets and Language", June 1997. Latest draft: <draft-avelstrand-charset-policy-00.txt>~~

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[print-mib-draft] Turner, R., "Printer MIB", work in progress, on the standards track as a draft standard: <draft-ietf-printmib-mib-info-02.txt>, ~~October 15~~January 22, 1999~~7~~.

[protomap] Bergman, R., "Job Submission Protocol Mapping Recommendations for the Job Monitoring MIB," work in progress as an informational RFC. See <draft-bergman-printmib-job-protomap-03~~1~~.txt>, ~~January 12~~February 10, 1998.

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

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

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

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

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

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

~~[rfc 2130] C. Weider, C. Preston, K. Simonsen, H. Alvestrand, R. Atkinson, M. Crispin, and P. Svanberg, "The Report of the IAB Character Set Workshop held 29 Feb-1 March, 1997", April 1997, RFC 2130.~~

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

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

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

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

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

[URI-spec] Berners-Lee, T., Fielding, R., Masinter, L., "Uniform Resource Identifiers (URI): Generic Syntax", RFC 2396, August 1998.~~Berners-Lee, T., Masinter, L., McCahill, M. , "Uniform Resource Locators (URL)", RFC 1738, December, 1994.~~

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

[UTF-8] F. Yergeau, "UTF-8, a transformation format of ~~Unicode and~~ ISO 10646", RFC ~~2044~~2279, ~~October 1996~~January 1998.

8 Notices

The IETF takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this

document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on the IETF's procedures with respect to rights in standards-track and standards-related documentation can be found in BCP-11[BCP-11]. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF Secretariat.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to practice this standard. Please address the information to the IETF Executive Director.

9 Author's Addresses

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

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

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

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

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

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

Harry Lewis
IBM Corporation
6300 Diagonal Hwy
Boulder, CO 80301

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

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

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

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

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

http://www.pwg.org/

Other Participants:

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

10 Change History

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

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

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

1. Added the Mirror table.

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

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

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

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

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

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

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

3. Removed the attributes from the Table of Contents

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

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

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

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

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

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

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

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

5. Changed "insure" to "ensure’.

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

11 INDEX

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