INTERNET-DRAFT R. Bergman
Dataproducts Corp.
T. Hastings (editor)
Xerox Corporation
S. Isaacson
Novell, Inc.
H. Lewis
IBM Corp.
February 19, 1999

Job Monitoring MIB - V1.0

<draft-ietf-printmib-job-monitor-08.txt>

Status of this Memo

Abstract

This document has been developed and approved by the Printer Working Group (PWG) as a PWG standard. It is intended to be distributed as an Informational RFC. This document provides a printer industry standard SNMP MIB for (1) monitoring the status and progress of print jobs (2) obtaining resource requirements before a job is processed, (3) monitoring resource consumption while a job is being processed and (4) collecting resource accounting data after the completion of a job. This MIB is intended to be implemented (1) in a printer or (2) in a server that supports one or more printers. Use of the object set is not limited to printing. However, support for services other than printing is outside the scope of this Job Monitoring MIB. Future extensions to this MIB may include, but are not limited to, fax machines and scanners.

TABLE OF CONTENTS

1 INTRODUCTION **5**

1.1 Types of Information in the MIB 5

1.2 Types of Job Monitoring Applications 7

2 TERMINOLOGY AND JOB MODEL **8**

2.1 System Configurations for the Job Monitoring MIB 11
2.1.1 Configuration 1 - client-printer 11
2.1.2 Configuration 2 - client-server-printer - agent in the server 12
2.1.3 Configuration 3 - client-server-printer - client monitors printer agent and server 13

3 MANAGED OBJECT USAGE **15**

3.1 Conformance Considerations 15
3.1.1 Conformance Terminology 15
3.1.2 Agent Conformance Requirements 15
3.1.2.1 MIB II System Group objects **16**
3.1.2.2 MIB II Interface Group objects **16**
3.1.2.3 Printer MIB objects **16**
3.1.3 Job Monitoring Application Conformance Requirements 16

3.2 The Job Tables and the Oldest Active and Newest Active Indexes 17

3.3 The Attribute Mechanism and the Attribute Table(s) 19
3.3.1 Conformance of Attribute Implementation 19
3.3.2 Useful, 'Unknown', and 'Other' Values for Objects and Attributes 20
3.3.3 Index Value Attributes 20
3.3.4 Data Sub-types and Attribute Naming Conventions 21
3.3.5 Single-Value (Row) Versus Multi-Value (MULTI-ROW) Attributes 22
3.3.6 Requested Objects and Attributes 22
3.3.7 Consumption Attributes 22
3.3.8 Attribute Specifications 23
3.3.9 Job State Reason bit definitions 43
3.3.9.1 JmJobStateReasons1TC specification **44**
3.3.9.2 JmJobStateReasons2TC specification **48**
3.3.9.3 JmJobStateReasons3TC specification **51**
3.3.9.4 JmJobStateReasons4TC specification **52**

3.4 Monitoring Job Progress 52

3.5 Job Identification 56

3.5.1 The Job Submission ID specifications 57

3.6 Internationalization Considerations 62
3.6.1 Text generated by the server or device 62
3.6.2 Text supplied by the job submitter 63
3.6.3 'DateAndTime' for representing the date and time 64

3.7 IANA and PWG Registration Considerations 64
3.7.1 PWG Registration of enums 64
3.7.1.1 Type 1 enumerations **65**
3.7.1.2 Type 2 enumerations **65**
3.7.1.3 Type 3 enumeration **65**
3.7.2 PWG Registration of type 2 bit values 66
3.7.3 PWG Registration of Job Submission Id Formats 66
3.7.4 PWG Registration of MIME types/sub-types for document-formats 66

3.8 Security Considerations 66
3.8.1 Read-Write objects 66
3.8.2 Read-Only Objects In Other User's Jobs 67

3.9 Notifications 67

4 MIB SPECIFICATION **67**

Textual conventions for this MIB module 69
JmUTF8StringTC 69
JmJobStringTC 69
JmNaturalLanguageTagTC 69
JmTimeStampTC 70
JmJobSourcePlatformTypeTC 70
JmFinishingTC 71
JmPrintQualityTC 72
JmPrinterResolutionTC 72
JmTonerEconomyTC 73
JmBooleanTC 73
JmMediumTypeTC 73
JmJobCollationTypeTC 75
JmJobSubmissionIDTypeTC 75
JmJobStateTC 76
JmAttributeTypeTC 79
JmJobServiceTypesTC 83
JmJobStateReasons1TC 84
JmJobStateReasons2TC 84
JmJobStateReasons3TC 85
JmJobStateReasons4TC 85

The General Group (MANDATORY) 86
jmGeneralJobSetIndex (Int32(1..32767)) 87
jmGeneralNumberOfActiveJobs (Int32(0..)) 87
jmGeneralOldestActiveJobIndex (Int32(0..)) 88

jmGeneralNewestActiveJobIndex (Int32(0..)) 88
jmGeneralJobPersistence (Int32(15..)) 89
jmGeneralAttributePersistence (Int32(15..)) 89
jmGeneralJobSetName (UTF8String63) 90

The Job ID Group (MANDATORY) 91
jmJobSubmissionID (OCTET STRING(SIZE(48))) 92
jmJobIDJobSetIndex (Int32(0..32767)) 93
jmJobIDJobIndex (Int32(0..)) 93

The Job Group (MANDATORY) 94
jmJobIndex (Int32(1..)) 95
jmJobState (JmJobStateTC) 95
jmJobStateReasons1 (JmJobStateReasons1TC) 96
jmNumberOfInterveningJobs (Int32(-2..)) 96
jmJobKOctetsPerCopyRequested (Int32(-2..)) 97
jmJobKOctetsProcessed (Int32(-2..)) 97
jmJobImpressionsPerCopyRequested (Int32(-2..)) 98
jmJobImpressionsCompleted (Int32(-2..)) 98
jmJobOwner (JobString63) 99

The Attribute Group (MANDATORY) 100
jmAttributeTypeIndex (JmAttributeTypeTC) 102
jmAttributeInstanceIndex (Int32(1..32767)) 102
jmAttributeValueAsInteger (Int32(-2..)) 103
jmAttributeValueAsOctets (Octets63) 104

5 APPENDIX A - IMPLEMENTING THE JOB LIFE CYCLE **107**

6 APPENDIX B - SUPPORT OF JOB SUBMISSION PROTOCOLS **108**

7 REFERENCES **108**

8 AUTHOR’S ADDRESSES **110**

9 CHANGE HISTORY **113**

9.1 Changes to produce version 1.0, dated February 19, 1999 113

9.2 Changes to produce version 1.2, dated October 2, 1998 113

9.3 Changes to produce version 1.1, dated October 1, 1998 114

10 INDEX **115**

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.

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.

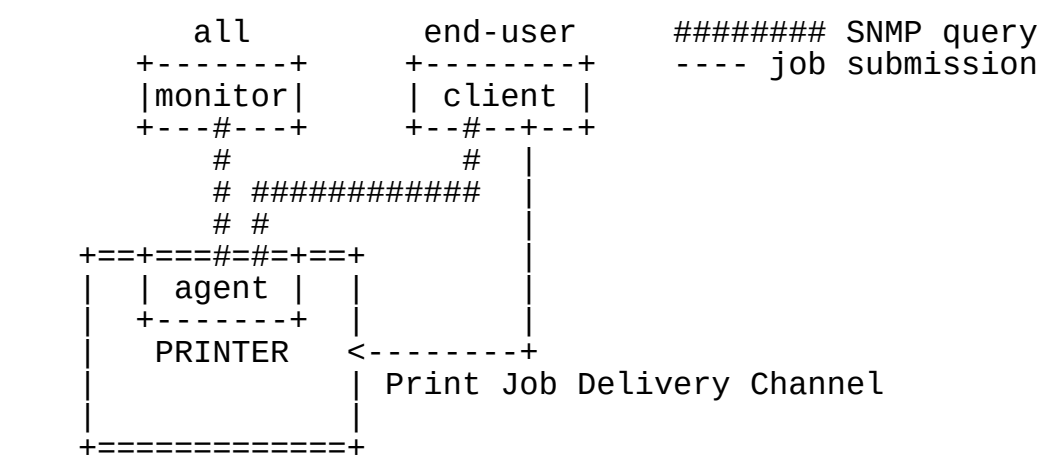

Figure 2-1 - Configuration 1 - client-printer - agent in the printer

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.

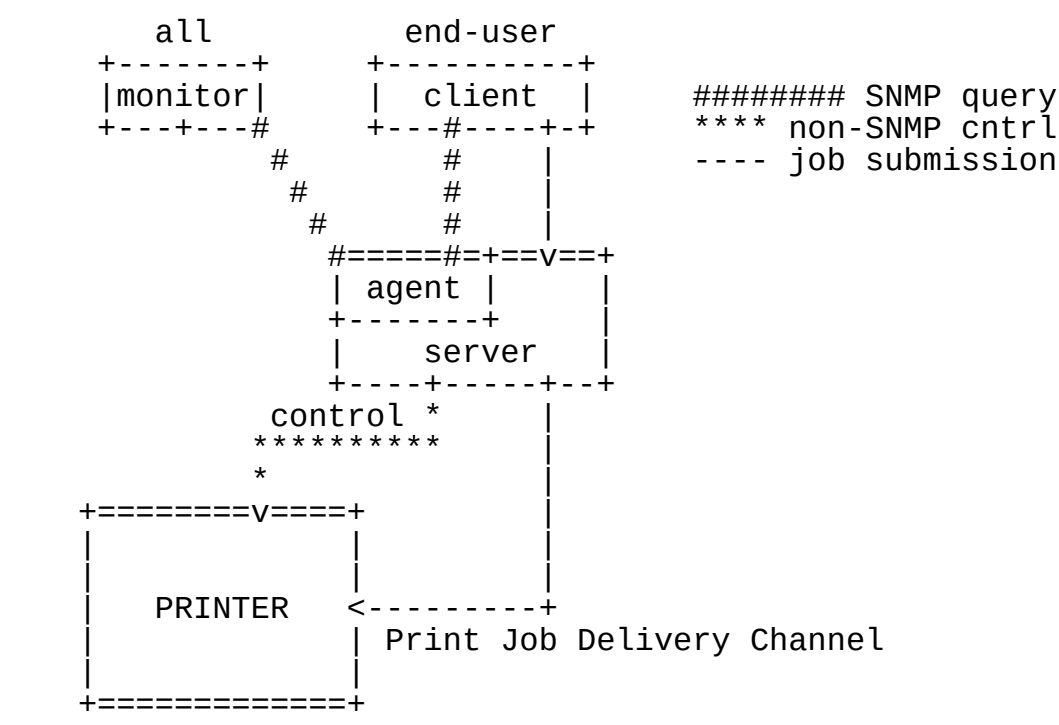

Figure 2-2 - Configuration 2 - client-server-printer - agent in the server

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.

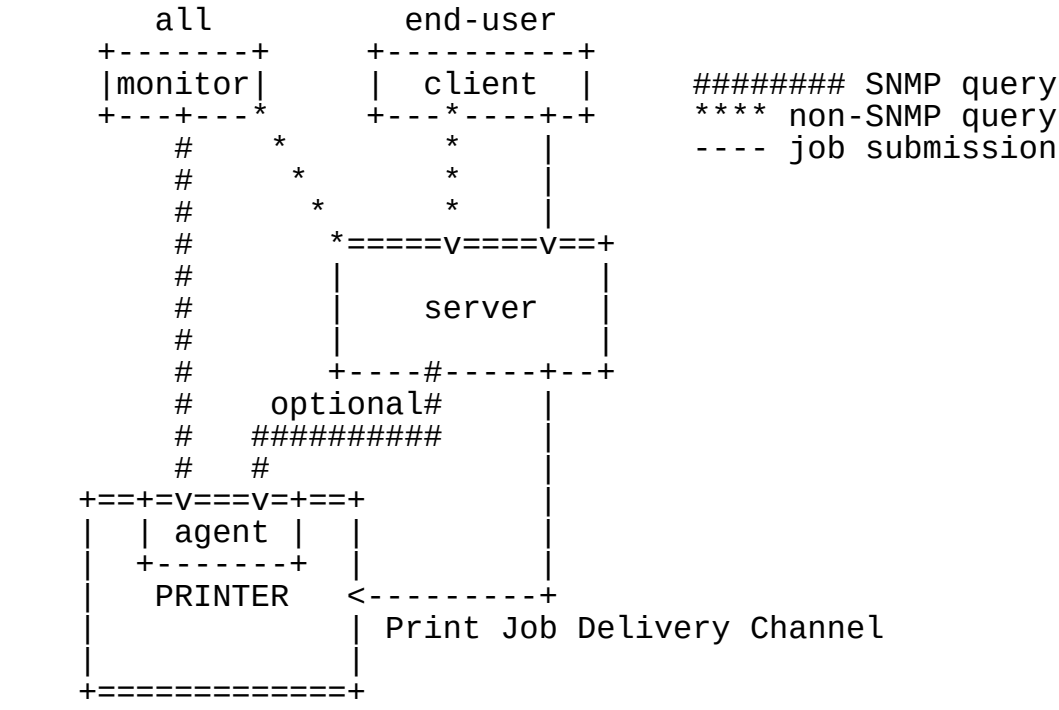

Figure 2-3 - Configuration 3 - client-server-printer - client monitors printer agent and server

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] 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.

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.

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.
++

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.

```
1019          ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
1020          + Job Identification attributes (20 - 49 decimal)
1021          +
1022          + The following attributes help an end user, a system
1023          + operator, or an accounting program identify a job.
1024          ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
1025
1026          jobURI(20),                          OCTET STRING(SIZE(0..63))
1027              OCTETS:  MULTI-ROW:  The job's Universal Resource
1028              Identifier (URI) [RFC1738].  See IPP [ipp-model] for
1029              example usage.
1030
1031              NOTE - The agent may be able to generate this value on each
1032              SNMP Get operation from smaller values, rather than having
1033              to store the entire URI.
1034
1035              If the URI exceeds 63 octets, the agent SHALL use multiple
1036              values, with the next 63 octets coming in the second value,
1037              etc.
1038
1039              NOTE - IPP [ipp-model] has a 1023-octet maximum length for
1040              a URI, though the URI standard itself and HTTP/1.1 specify
1041              no maximum length.
1042
1043          jobAccountName(21),                  OCTET STRING(SIZE(0..63))
1044              OCTETS:  Arbitrary binary information which MAY be coded
1045              character set data or encrypted data supplied by the
1046              submitting user for use by accounting services to allocate
1047              or categorize charges for services provided, such as a
1048              customer account name or number.
1049
1050              NOTE: This attribute NEED NOT be printable characters.
1051
1052          serverAssignedJobName(22),           JmJobStringTC (SIZE(0..63))
1053              OCTETS:  Configuration 3 only:  The human readable string
1054              name, number, or ID of the job as assigned by the server
1055              that submitted the job to the device that the agent is
1056              providing access to with this MIB.
1057
1058              NOTE - This attribute is intended for enabling a user to
1059              find his/her job that a server submitted to a device when
1060              either the client does not support the jmJobSubmissionID or
1061              the server does not pass the jmJobSubmissionID through to
1062              the device.
```

```
1063
1064          jobName(23),                        JmJobStringTC (SIZE(0..63))
1065              OCTETS:  The human readable string name of the job as
1066              assigned by the submitting user to help the user
1067              distinguish between his/her various jobs.  This name does
1068              not need to be unique.
1069
1070              This attribute is intended for enabling a user or the
1071              user's application to convey a job name that MAY be printed
1072              on a start sheet, returned in a query result, or used in
1073              notification or logging messages.
1074
1075              In order to assist users to find their jobs for job
1076              submission protocols that don't supply a jmJobSubmissionID,
1077              the agent SHOULD maintain the jobName attribute for the
1078              time specified by the jmGeneralJobPersistence object,
1079              rather than the (shorter) jmGeneralAttributePersistence
1080              object.
1081
1082              If this attribute is not specified when the job is
1083              submitted, no job name is assumed, but implementation
1084              specific defaults are allowed, such as the value of the
1085              documentName attribute of the first document in the job or
1086              the fileName attribute of the first document in the job.
1087
1088              The jobName attribute is distinguished from the jobComment
1089              attribute, in that the jobName attribute is intended to
1090              permit the submitting user to distinguish between different
1091              jobs that he/she has submitted.  The jobComment attribute
1092              is intended to be free form additional information that a
1093              user might wish to use to communicate with himself/herself,
1094              such as a reminder of what to do with the results or to
1095              indicate a different set of input parameters were tried in
1096              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.

```
1234
1235          documentFormat(38),                PrtInterpreterLangFamilyTC
1236                                             AND/OR
1237                                             OCTET STRING(SIZE(0..63))
1238              INTEGER:  MULTI-ROW:  The interpreter language family
1239              corresponding to the Printer MIB[print-mib]
1240              prtInterpreterLangFamily object, that this job
1241              requires/uses.  A document or a job MAY use more than one
1242              PDL or control language.
1243
1244              AND/OR
1245
1246              OCTETS:  MULTI-ROW:  The document format registered as a
1247              media type[iana-media-types], i.e., the name of the MIME
1248              content-type/subtype.  Examples: 'application/postscript',
1249              'application/vnd.hp-PCL', 'application/pdf', 'text/plain'
1250              (US-ASCII SHALL be assumed), 'text/plain; charset=iso-8859-
1251              1', and 'application/octet-stream'.  The IPP 'document-
1252              format' job attribute uses these same values with the same
1253              semantics.  See the IPP [ipp-model] 'mimeMediaType'
1254              attribute syntax and the document-format attribute for
1255              further examples and explanation.
1256
1257          ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
1258          + Job Parameter attributes (50 - 67 decimal)
1259          +
1260          + The following attributes represent input parameters
1261          + supplied by the submitting client in the job submission
1262          + protocol.
1263          ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
1264
1265          jobPriority(50),                   Integer32 (-2..100)
1266              INTEGER:  The priority for scheduling the job.  It is used
1267              by servers and devices that employ a priority-based
1268              scheduling algorithm.
1269
1270              A higher value specifies a higher priority.  The value 1 is
1271              defined to indicate the lowest possible priority (a job
1272              which a priority-based scheduling algorithm SHALL pass over
1273              in favor of higher priority jobs).  The value 100 is
1274              defined to indicate the highest possible priority.
1275              Priority is expected to be evenly or 'normally' distributed
1276              across this range.  The mapping of vendor-defined priority
1277              over this range is implementation-specific.  -2 indicates
1278              unknown.
```

```
1279
1280          jobProcessAfterDateAndTime(51),   DateAndTime (SNMPv2-TC)
1281              OCTETS:  The calendar date and time of day after which the
1282              job SHALL become a candidate to be scheduled for
1283              processing.  If the value of this attribute is in the
1284              future, the server SHALL set the value of the job's
1285              jmJobState object to pendingHeld and add the
1286              jobProcessAfterSpecified bit value to the job's
1287              jmJobStateReasons1 object.  When the specified date and
1288              time arrives, the server SHALL remove the
1289              jobProcessAfterSpecified bit value from the job's
1290              jmJobStateReasons1 object and, if no other reasons remain,
1291              SHALL change the job's jmJobState object to pending.
1292
1293          jobHold(52),                        JmBooleanTC
1294              INTEGER:  If the value is 'true(4)', a client has
1295              explicitly specified that the job is to be held until
1296              explicitly released.  Until the job is explicitly released
1297              by a client, the job SHALL be in the pendingHeld state with
1298              the jobHoldSpecified value in the jmJobStateReasons1
1299              attribute.
1300
1301          jobHoldUntil(53),                   JmJobStringTC (SIZE(0..63))
1302              OCTETS:  The named time period during which the job SHALL
1303              become a candidate for processing, such as 'evening',
1304              'night', 'weekend', 'second-shift', 'third-shift', etc.,
1305              (supported values configured by the system administrator).
1306              See IPP [ipp-model] for the standard keyword values.  Until
1307              that time period arrives, the job SHALL be in the
1308              pendingHeld state with the jobHoldUntilSpecified value in
1309              the jmJobStateReasons1 object.  The value 'no-hold' SHALL
1310              indicate explicitly that no time period has been specified;
1311              the absence of this attribute SHALL indicate implicitly
1312              that no time period has been specified.
1313
1314          outputBin(54),                      Integer32 (0..2147483647)
1315                                              AND/OR
1316                                              JmJobStringTC (SIZE(0..63))
1317              INTEGER:  MULTI-ROW:  The output subunit index in the
1318              Printer MIB[print-mib]
1319
1320              AND/OR
1321
1322              OCTETS:  MULTI-ROW:  the name or number (represented as
1323              ASCII digits) of the output bin to which all or part of the
1324              job is placed in.
1325
1326          sides(55),                          Integer32 (-2..2)
1327              INTEGER:  MULTI-ROW:  The number of sides, '1' or '2', that
1328              any document in this job requires/used.
```

```
1329
1330          finishing(56),                    JmFinishingTC
1331              INTEGER:  MULTI-ROW:  Type of finishing that any document
1332              in this job requires/used.
1333
1334
1335          ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
1336          + Image Quality attributes (requested and consumed) (70 - 87)
1337          +
1338          + For devices that can vary the image quality.
1339          ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
1340
1341          printQualityRequested(70),        JmPrintQualityTC
1342              INTEGER:  MULTI-ROW:  The print quality selection requested
1343              for a document in the job for printers that allow quality
1344              differentiation.
1345
1346          printQualityUsed(71),             JmPrintQualityTC
1347              INTEGER:  MULTI-ROW:  The print quality selection actually
1348              used by a document in the job for printers that allow
1349              quality differentiation.
1350
1351          printerResolutionRequested(72),   JmPrinterResolutionTC
1352              OCTETS:  MULTI-ROW:  The printer resolution requested for a
1353              document in the job for printers that support resolution
1354              selection.
1355
1356          printerResolutionUsed(73),        JmPrinterResolutionTC
1357              OCTETS:  MULTI-ROW:  The printer resolution actually used
1358              by a document in the job for printers that support
1359              resolution selection.
1360
1361          tonerEcomonyRequested(74),        JmTonerEconomyTC
1362              INTEGER:  MULTI-ROW:  The toner economy selection requested
1363              for documents in the job for printers that allow toner
1364              economy differentiation.
1365
1366          tonerEcomonyUsed(75),             JmTonerEconomyTC
1367              INTEGER:  MULTI-ROW:  The toner economy selection actually
1368              used by documents in the job for printers that allow toner
1369              economy differentiation.
1370
1371          tonerDensityRequested(76)         Integer32 (-2..100)
1372              INTEGER:  MULTI-ROW:  The toner density requested for a
1373              document in this job for devices that can vary toner
1374              density levels.  Level 1 is the lowest density and level
1375              100 is the highest density level.  Devices with a smaller
1376              range, SHALL map the 1-100 range evenly onto the
1377              implemented range.
```

```
1378
1379          tonerDensityUsed(77),              Integer32 (-2..100)
1380              INTEGER:  MULTI-ROW:  The toner density used by documents
1381              in this job for devices that can vary toner density levels.
1382              Level 1 is the lowest density and level 100 is the highest
1383              density level.  Devices with a smaller range, SHALL map the
1384              1-100 range evenly onto the implemented range.
1385
1386          ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
1387          + Job Progress attributes (requested and consumed) (90-109)
1388          +
1389          + Pairs of these attributes can be used by monitoring
1390          + applications to show an indication of relative progress
1391          + to users.  See section 3.4, entitled:
1392          + 'Monitoring Job Progress'.
1393          ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
1394
1395          jobCopiesRequested(90),            Integer32 (-2..2147483647)
1396              INTEGER:  The number of copies of the entire job that are
1397              to be produced.
1398
1399          jobCopiesCompleted(91),            Integer32 (-2..2147483647)
1400              INTEGER:  The number of copies of the entire job that have
1401              been completed so far.
1402
1403          documentCopiesRequested(92),       Integer32 (-2..2147483647)
1404              INTEGER:  The total count of the number of document copies
1405              requested for the job as a whole.  If there are documents
1406              A, B, and C, and document B is specified to produce 4
1407              copies, the number of document copies requested is 6 for
1408              the job.
1409
1410              This attribute SHALL be used only when a job has multiple
1411              documents.  The jobCopiesRequested attribute SHALL be used
1412              when the job has only one document.
1413
1414          documentCopiesCompleted(93),       Integer32 (-2..2147483647)
1415              INTEGER:  The total count of the number of document copies
1416              completed so far for the job as a whole.  If there are
1417              documents A, B, and C, and document B is specified to
1418              produce 4 copies, the number of document copies starts a 0
1419              and runs up to 6 for the job as the job processes.
1420
1421              This attribute SHALL be used only when a job has multiple
1422              documents.  The jobCopiesCompleted attribute SHALL be used
1423              when the job has only one document.
```

```
1424
1425          jobKOctetsTransferred(94),        Integer32 (-2..2147483647)
1426              INTEGER:  The number of K (1024) octets transferred to the
1427              server or device to which the agent is providing access.
1428              This count is independent of the number of copies of the
1429              job or documents that will be produced, but it is only a
1430              measure of the number of bytes transferred to the server or
1431              device.
1432
1433              The agent SHALL round the actual number of octets
1434              transferred up to the next higher K.  Thus 0 octets SHALL
1435              be represented as '0', 1-1024 octets SHALL BE represented
1436              as '1', 1025-2048 SHALL be '2', etc.  When the job
1437              completes, the values of the jmJobKOctetsPerCopyRequested
1438              object and the jobKOctetsTransferred attribute SHALL be
1439              equal.
1440
1441              NOTE - The jobKOctetsTransferred can be used with the
1442              jmJobKOctetsPerCopyRequested object in order to produce a
1443              relative indication of the progress of the job for agents
1444              that do not implement the jmJobKOctetsProcessed object.
1445
1446          sheetCompletedCopyNumber(95),     Integer32 (-2..2147483647)
1447              INTEGER:  The number of the copy being stacked for the
1448              current document.  This number starts at 0, is set to 1
1449              when the first sheet of the first copy for each document is
1450              being stacked and is equal to n where n is the nth sheet
1451              stacked in the current document copy.  See section 3.4 ,
1452              entitled 'Monitoring Job Progress'.
1453
1454          sheetCompletedDocumentNumber(96), Integer32 (-2..2147483647)
1455              INTEGER:  The ordinal number of the document in the job
1456              that is currently being stacked.  This number starts at 0,
1457              increments to 1 when the first sheet of the first document
1458              in the job is being stacked, and is equal to n where n is
1459              the nth document in the job, starting with 1.
1460
1461              Implementations that only support one document jobs SHOULD
1462              NOT implement this attribute.
1463
1464          jobCollationType(97),             JmJobCollationTypeTC
1465              INTEGER:  The type of job collation. See also Section 3.4,
1466              entitled 'Monitoring Job Progress'.
1467
```

```
        ++++++++++++++++++++++++++++++++++++++++++++++++++++++++
        + 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.
        ++++++++++++++++++++++++++++++++++++++++++++++++++++++++

        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)).
```

```
1515
1516          highlightColorImpressionsCompleted(115),
1517                                              Integer32 (-2..2147483647)
1518              INTEGER:  The number of highlight color impressions
1519              completed by the device for this job so far.  For printing,
1520              the impressions completed includes interpreting, marking,
1521              and stacking the output.  For other types of job services,
1522              the number of impressions completed includes the number of
1523              impressions processed.  Highlight color impressions are
1524              typically defined as those requiring black plus one other
1525              colorant, but this MAY vary by implementation.  In any
1526              case, the value of this attribute counts by 1 for each side
1527              that has highlight color (and the other impression counters
1528              are incremented, except
1529              fullColorImpressionsCompleted(114)).
1530
1531          ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
1532          + Page attributes (130 - 149 decimal)
1533          +
1534          + See the definition of 'impression', 'sheet', and 'page'
1535          + in Section 2.
1536          ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
1537
1538          pagesRequested(130),                Integer32 (-2..2147483647)
1539              INTEGER:  The number of logical pages requested by the job
1540              to be processed.
1541
1542          pagesCompleted(131),                Integer32 (-2..2147483647)
1543              INTEGER:  The number of logical pages completed for this
1544              job so far.
1545
1546              For implementations where multiple copies are produced by
1547              the interpreter with only a single pass over the data, the
1548              final value SHALL be equal to the value of the
1549              pagesRequested object.  For implementations where multiple
1550              copies are produced by the interpreter by processing the
1551              data for each copy, the final value SHALL be a multiple of
1552              the value of the pagesRequested object.
1553
1554              NOTE - See the impressionsCompletedCurrentCopy and
1555              pagesCompletedCurrentCopy attributes for attributes that
1556              are reset on each document copy.
1557
1558              NOTE - The pagesCompleted object can be used with the
1559              pagesRequested object to provide an indication of the
1560              relative progress of the job, provided that the
1561              multiplicative factor is taken into account for some
1562              implementations of multiple copies.
```

```
1563
1564          pagesCompletedCurrentCopy(132),   Integer32 (-2..2147483647)
1565              INTEGER:  The number of logical pages completed for the
1566              current copy of the document so far.  This value SHALL be
1567              reset to 0 for each document in the job and for each
1568              document copy.
1569
1570          ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
1571          + Sheet attributes (150 - 169 decimal)
1572          +
1573          + See the definition of 'impression', 'sheet', and 'page'
1574          + in Section 2.
1575          ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
1576
1577          sheetsRequested(150),             Integer32 (-2..2147483647)
1578              INTEGER:  The total number of medium sheets requested to be
1579              produced for this job.
1580
1581              Unlike the jmJobKOctetsPerCopyRequested and
1582              jmJobImpressionsPerCopyRequested attributes, the
1583              sheetsRequested(150) attribute SHALL include the
1584              multiplicative factor contributed by the number of copies
1585              and so is the total number of sheets to be produced by the
1586              job, as opposed to the size of the document(s) submitted.
1587
1588          sheetsCompleted(151),             Integer32 (-2..2147483647)
1589              INTEGER:  The total number of medium sheets that have
1590              completed marking and stacking for the entire job so far
1591              whether those sheets have been processed on one side or on
1592              both.
1593
1594          sheetsCompletedCurrentCopy(152),  Integer32 (-2..2147483647)
1595              INTEGER:  The number of medium sheets that have completed
1596              marking and stacking for the current copy of a document in
1597              the job so far whether those sheets have been processed on
1598              one side or on both.
1599
1600              The value of this attribute SHALL be 0 before the job
1601              starts processing and SHALL be reset to 1 after the first
1602              sheet of each document and document copy in the job is
1603              processed and stacked.
1604
```

```
1605          ++++++++++++++++++++++++++++++++++++++++++++++++++++++++
1606          + Resources attributes (requested and consumed) (170 - 189)
1607          +
1608          + Pairs of these attributes can be used by monitoring
1609          + applications to show an indication of relative usage to
1610          + users, i.e., a 'thermometer'.
1611          ++++++++++++++++++++++++++++++++++++++++++++++++++++++++
1612
1613          mediumRequested(170),              JmMediumTypeTC
1614                                             AND/OR
1615                                             JmJobStringTC (SIZE(0..63))
1616              INTEGER:  MULTI-ROW:  The type
1617              AND/OR
1618              OCTETS:  MULTI-ROW:  the name of the medium that is
1619              required by the job.
1620
1621              NOTE - The name (JmJobStringTC) values correspond to the
1622              name values of the prtInputMediaName object in the Printer
1623              MIB [print-mib] and the name, size, and input tray values
1624              of the IPP 'media' attribute [ipp-model].
1625
1626          mediumConsumed(171),               Integer32 (-2..2147483647)
1627                                             AND
1628                                             JmJobStringTC (SIZE(0..63))
1629              INTEGER:  MULTI-ROW:  The number of sheets
1630              AND
1631              OCTETS:  MULTI-ROW:  the name of the medium that has been
1632              consumed so far whether those sheets have been processed on
1633              one side or on both.
1634
1635              This attribute SHALL have both Integer32 and OCTET STRING
1636              (represented as  JmJobStringTC) values.
1637
1638              NOTE - The name (JmJobStringTC) values correspond to the
1639              name values of the prtInputMediaName object in the Printer
1640              MIB [print-mib] and the name, size, and input tray values
1641              of the IPP 'media' attribute [ipp-model].
1642
1643          colorantRequested(172),            Integer32 (-2..2147483647)
1644                                             AND/OR
1645                                             JmJobStringTC (SIZE(0..63))
1646              INTEGER:  MULTI-ROW:  The index (prtMarkerColorantIndex) in
1647              the Printer MIB[print-mib]
1648              AND/OR
1649              OCTETS:  MULTI-ROW:  the name of the colorant requested.
1650
1651              NOTE - The name (JmJobStringTC) values correspond to the
1652              name values of the prtMarkerColorantValue object in the
1653              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.

```
1705          ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
1706          + Time attributes (set by server or device) (190 - 209 decimal)
1707          +
1708          + This section of attributes are ones that are set by the
1709          + server or device that accepts jobs.  Two forms of time are
1710          + provided.  Each form is represented in a separate attribute.
1711          + See section 3.1.2 and section 3.1.3 for the
1712          + conformance requirements for time attribute for agents and
1713          + monitoring applications, respectively.  The two forms are:
1714          +
1715          + 'DateAndTime' is an 8 or 11 octet binary encoded year,
1716          + month, day, hour, minute, second, deci-second with
1717          + optional offset from UTC.  See SNMPv2-TC [SMIv2-TC].
1718          +
1719          + NOTE: 'DateAndTime' is not printable characters; it is
1720          + binary.
1721          +
1722          + 'JmTimeStampTC' is the time of day measured in the number of
1723          + seconds since the system was booted.
1724          ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
1725
1726          jobSubmissionToServerTime(190),   JmTimeStampTC
1727                                            AND/OR
1728                                            DateAndTime
1729              INTEGER:  Configuration 3 only:  The time
1730              AND/OR
1731              OCTETS:  the date and time that the job was submitted to
1732              the server (as distinguished from the device which uses
1733              jobSubmissionTime).
1734
1735          jobSubmissionTime(191),           JmTimeStampTC
1736                                            AND/OR
1737                                            DateAndTime
1738              INTEGER:  Configurations 1, 2, and 3:  The time
1739              AND/OR
1740              OCTETS:  the date and time that the job was submitted to
1741              the server or device to which the agent is providing
1742              access.
1743
1744          jobStartedBeingHeldTime(192),     JmTimeStampTC
1745                                            AND/OR
1746                                            DateAndTime
1747              INTEGER:  The time
1748              AND/OR
1749              OCTETS:  the date and time that the job last entered the
1750              pendingHeld state.  If the job has never entered the
1751              pendingHeld state, then the value SHALL be '0' or the
1752              attribute SHALL not be present in the table.
```

```
1753
1754          jobStartedProcessingTime(193),    JmTimeStampTC
1755                                            AND/OR
1756                                            DateAndTime
1757              INTEGER:  The time
1758              AND/OR
1759              OCTETS:  the date and time that the job started processing.
1760
1761          jobCompletionTime(194),           JmTimeStampTC
1762                                            AND/OR
1763                                            DateAndTime
1764              INTEGER:  The time
1765              AND/OR
1766              OCTETS:  the date and time that the job entered the
1767              completed, canceled, or aborted state.
1768
1769          jobProcessingCPUTime(195)         Integer32 (-2..2147483647)
1770              UNITS     'seconds'
1771              INTEGER:  The amount of CPU time in seconds that the job
1772              has been in the processing state.  If the job enters the
1773              processingStopped state, that elapsed time SHALL not be
1774              included.  In other words, the jobProcessingCPUTime value
1775              SHOULD be relatively repeatable when the same job is
1776              processed again on the same device.
```

1777 3.3.9 Job State Reason bit definitions

1778 The JmJobStateReasons*N*TC (*N*=1..4) textual-conventions are used with the
1779 jmJobStateReasons1 object and jobStateReasonsN (*N*=2..4), respectively,
1780 to provide additional information regarding the current jmJobState
1781 object value. These values MAY be used with any job state or states
1782 for which the reason makes sense.

1783 NOTE - While values cannot be added to the jmJobState object without
1784 impacting deployed clients that take actions upon receiving jmJobState
1785 values, it is the intent that additional JmJobStateReasons*N*TC enums can
1786 be defined and registered without impacting such deployed clients. In
1787 other words, the jmJobStateReasons1 object and jobStateReasons*N*
1788 attributes are intended to be extensible.

1789 NOTE - The Job Monitoring MIB contains a superset of the IPP
1790 values[ipp-model] for the IPP 'job-state-reasons' attribute, since the
1791 Job Monitoring MIB is intended to cover other job submission protocols
1792 as well. Also some of the names of the reasons have been changed from
1793 'printer' to 'device', since the Job Monitoring MIB is intended to
1794 cover additional types of devices, including input devices, such as
1795 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 [RFC1179] which includes the host name in the job submission protocol.
octets 41-48: The US-ASCII 8-decimal-digit leading zero representation of the job id generated by the submitting server (configuration 3) or the client (configuration 1 and 2), such as in the LPD protocol.
This format is reserved for clients.

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

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

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

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

'E' IPDS on the MVS or VSE platform

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

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

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

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

3.6 Internationalization Considerations

This section describes the internationalization considerations included in this MIB.

3.6.1 Text generated by the server or device

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

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

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

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

The text contained in the processingMessage(6) attribute is generated by the server/device. The natural language for the processingMessage(6) attribute is identified by the processingMessageNaturalLangTag(7) attribute. The processingMessageNaturalLangTag(7) attribute uses the JmNaturalLanguageTagTC textual convention which SHALL conform to the language tag mechanism specified in RFC 1766 [RFC1766]. 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 "9902190000Z"
    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: 1.0"
    ::= { enterprises pwg(2699)  mibs(1)  jobmonMIB(1) }
```

```
2738
2739
2740   -- Textual conventions for this MIB module
2741
2742   JmUTF8StringTC ::= TEXTUAL-CONVENTION
2743       DISPLAY-HINT "255a"
2744       STATUS      current
2745       DESCRIPTION
2746           "To facilitate internationalization, this TC represents
2747           information taken from the ISO/IEC IS 10646-1 character set,
2748           encoded as an octet string using the UTF-8 character encoding
2749           scheme.
2750
2751           See section 3.6.1, entitled: 'Text generated by the server or
2752           device'."
2753       SYNTAX      OCTET STRING (SIZE (0..63))
2754
2755
2756
2757
2758   JmJobStringTC ::= TEXTUAL-CONVENTION
2759       STATUS      current
2760       DESCRIPTION
2761           "To facilitate internationalization, this TC represents
2762           information using any coded character set registered by IANA as
2763           specified in section 3.7.  While it is recommended that the
2764           coded character set be UTF-8 [UTF-8], the actual coded
2765           character set SHALL be indicated by the value of the
2766           jobCodedCharSet(8) attribute for the job.
2767
2768           See section 3.6.2, entitled: 'Text supplied by the job
2769           submitter'."
2770       SYNTAX      OCTET STRING (SIZE (0..63))
2771
2772
2773
2774
2775   JmNaturalLanguageTagTC  ::= TEXTUAL-CONVENTION
2776       STATUS      current
2777       DESCRIPTION
2778           "An IETF RFC 1766-compliant 'language tag', with zero or more
2779           sub-tags that identify a natural language.  While RFC 1766
2780           specifies that the US-ASCII values are case-insensitive, this
2781           MIB specification requires that all characters SHALL be lower
2782           case in order to simplify comparing by management applications.
2783
2784           See section 3.6.1, entitled: 'Text generated by the server or
2785           device' and section 3.6.2, entitled: 'Text supplied by the job
2786           submitter'."
2787       SYNTAX      OCTET STRING (SIZE (0..63))
```

```
2788
2789
2790   JmTimeStampTC ::= TEXTUAL-CONVENTION
2791       STATUS      current
2792       DESCRIPTION
2793           "The simple time at which an event took place.  The units are
2794           in seconds since the system was booted.
2795
2796           NOTE - JmTimeStampTC is defined in units of seconds, rather
2797           than 100ths of seconds, so as to be simpler for agents to
2798           implement (even if they have to implement the 100ths of a
2799           second to comply with implementing sysUpTime in MIB-II[mib-
2800           II].)
2801
2802           NOTE - JmTimeStampTC is defined as an Integer32 so that it can
2803           be used as a value of an attribute, i.e., as a value of the
2804           jmAttributeValueAsInteger object.  The TimeStamp textual-
2805           convention defined in SNMPv2-TC [SMIv2-TC] is defined as an
2806           APPLICATION 3 IMPLICIT INTEGER tag, not an Integer32 which is
2807           defined in SNMPv2-SMI [SMIv2-TC] as UNIVERSAL 2 IMPLICIT
2808           INTEGER, so cannot be used in this MIB as one of the values of
2809           jmAttributeValueAsInteger."
2810       SYNTAX      INTEGER (0..2147483647)
2811
2812
2813
2814
2815   JmJobSourcePlatformTypeTC ::= TEXTUAL-CONVENTION
2816       STATUS      current
2817       DESCRIPTION
2818           "The source platform type that can submit jobs to servers or
2819           devices in any of the 3 configurations.
2820
2821           This is a type 2 enumeration.  See Section 3.7.1.2.  See also
2822           IANA operating-system-names registry."
2823       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
2824       }
```

```
2825
2826
2827   JmFinishingTC ::= TEXTUAL-CONVENTION
2828       STATUS      current
2829       DESCRIPTION
2830           "The type of finishing operation.
2831
2832           These values are the same as the enum values of the IPP
2833           'finishings' attribute.  See Section 3.7.1.2.
2834
2835           other(1),
2836               Some other finishing operation besides one of the specified
2837               or registered values.
2838
2839           unknown(2),
2840               The finishing is unknown.
2841
2842           none(3),
2843               Perform no finishing.
2844
2845           staple(4),
2846               Bind the document(s) with one or more staples. The exact
2847               number and placement of the staples is site-defined.
2848
2849           punch(5),
2850               Holes are required in the finished document. The exact
2851               number and placement of the holes is site-defined.  The
2852               punch specification MAY be satisfied (in a site- and
2853               implementation-specific manner) either by
2854               drilling/punching, or by substituting pre-drilled media.
2855
2856           cover(6),
2857               Select a non-printed (or pre-printed) cover for the
2858               document. This does not supplant the specification of a
2859               printed cover (on cover stock medium) by the document
2860               itself.
2861
2862           bind(7)
2863               Binding is to be applied to the document; the type and
2864               placement of the binding is product-specific.
2865
2866           This is a type 2 enumeration.  See Section 3.7.1.2."
2867       SYNTAX      INTEGER {
2868           other(1),
2869           unknown(2),
2870           none(3),
2871           staple(4),
2872           punch(5),
2873           cover(6),
2874           bind(7)
2875       }
```

```
2876
2877
2878   JmPrintQualityTC ::= TEXTUAL-CONVENTION
2879       STATUS      current
2880       DESCRIPTION
2881           "Print quality settings.
2882
2883           These values are the same as the enum values of the IPP 'print-
2884           quality' attribute.  See Section 3.7.1.2.
2885
2886           This is a type 2 enumeration.  See Section 3.7.1.2."
2887       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.
2888       }
2889
2890
2891
2892   JmPrinterResolutionTC ::= TEXTUAL-CONVENTION
2893       STATUS      current
2894       DESCRIPTION
2895           "Printer resolutions.
2896
2897           Nine octets consisting of two 4-octet SIGNED-INTEGERs followed
2898           by a SIGNED-BYTE.  The values are the same as those specified
2899           in the Printer MIB [printmib]. The first SIGNED-INTEGER
2900           contains the value of prtMarkerAddressabilityXFeedDir.  The
2901           second SIGNED-INTEGER contains the value of
2902           prtMarkerAddressabilityFeedDir.  The SIGNED-BYTE contains the
2903           value of prtMarkerAddressabilityUnit.
2904
2905           Note: the latter value is either 3 (tenThousandsOfInches) or 4
2906           (micrometers) and the addressability is in 10,000 units of
2907           measure. Thus the SIGNED-INTEGERs represent integral values in
2908           either dots-per-inch or dots-per-centimeter.
2909
2910           The syntax is the same as the IPP 'printer-resolution'
2911           attribute.  See Section 3.7.1.2."
2912       SYNTAX      OCTET STRING (SIZE(9))
2913
2914
```

```
2915
2916
2917   JmTonerEconomyTC ::= TEXTUAL-CONVENTION
2918       STATUS      current
2919       DESCRIPTION
2920           "Toner economy settings.
2921
2922           This is a type 2 enumeration.  See Section 3.7.1.2."
2923       SYNTAX      INTEGER {
                        unknown(2),    --  unknown.
                        off(3),        --  Off.  Normal.  Use full toner.
                        on(4)          --  On.  Use less toner than normal.
2924       }
2925
2926
2927
2928   JmBooleanTC ::= TEXTUAL-CONVENTION
2929       STATUS      current
2930       DESCRIPTION
2931           "Boolean true or false value.
2932
2933           This is a type 2 enumeration.  See Section 3.7.1.2."
2934       SYNTAX      INTEGER {
                        unknown(2),    --  unknown.
                        false(3),      --  FALSE.
                        true(4)        --  TRUE.
2935       }
2936
2937
2938
2939   JmMediumTypeTC ::= TEXTUAL-CONVENTION
2940       STATUS      current
2941       DESCRIPTION
2942           "Identifies the type of medium.
2943
2944           other(1),
2945               The type is neither one of the values listed in this
2946               specification nor a registered value.
2947
2948           unknown(2),
2949               The type is not known.
2950
2951           stationery(3),
2952               Separately cut sheets of an opaque material.
2953
2954           transparency(4),
2955               Separately cut sheets of a transparent material.
2956
2957           envelope(5),
2958               Envelopes that can be used for conventional mailing
2959               purposes.
```

```
2960
2961          envelopePlain(6),
2962              Envelopes that are not preprinted and have no windows.
2963
2964          envelopeWindow(7),
2965              Envelopes that have windows for addressing purposes.
2966
2967          continuousLong(8),
2968              Continuously connected sheets of an opaque material
2969              connected along the long edge.
2970
2971          continuousShort(9),
2972              Continuously connected sheets of an opaque material
2973              connected along the short edge.
2974
2975          tabStock(10),
2976              Media with tabs.
2977
2978          multiPartForm(11),
2979              Form medium composed of multiple layers not pre-attached to
2980              one another;  each sheet MAY be drawn separately from an
2981              input source.
2982
2983          labels(12),
2984              Label-stock.
2985
2986          multiLayer(13)
2987              Form medium composed of multiple layers which are pre-
2988              attached to one another, e.g. for use with impact printers.
2989
2990          This is a type 2 enumeration.  See Section 3.7.1.2.  These enum
2991          values correspond to the keyword name strings of the
2992          prtInputMediaType object in the Printer MIB [print-mib].  There
2993          is no printer description attribute in IPP/1.0 that represents
2994          these values."
2995      SYNTAX      INTEGER {
2996          other(1),
2997          unknown(2),
2998          stationery(3),
2999          transparency(4),
3000          envelope(5),
3001          envelopePlain(6),
3002          envelopeWindow(7),
3003          continuousLong(8),
3004          continuousShort(9),
3005          tabStock(10),
3006          multiPartForm(11),
3007          labels(12),
3008          multiLayer(13)
3009      }
3010
```

```
3011
3012
3013   JmJobCollationTypeTC ::= TEXTUAL-CONVENTION
3014       STATUS      current
3015       DESCRIPTION
3016           "This value is the type of job collation.  Implementations that
3017           don't support multiple documents or don't support multiple
3018           copies SHALL NOT support the uncollatedDocuments(5) value.
3019
3020           This is a type 2 enumeration.  See Section 3.7.1.2. See also
3021           Section 3.4, entitled 'Monitoring Job Progress'."
3022       SYNTAX      INTEGER {
3023           other(1),
3024           unknown(2),
3025           uncollatedSheets(3),    -- sheets within each document copy
3026                                   -- are not collated: 1 1 ..., 2 2 ...,
3027                                   -- No corresponding value of IPP
3028                                   -- "multiple-document-handling"
3029           collatedDocuments(4),   -- internal collated sheets,
3030                                   -- documents: A, B, A, B, ...
3031                                   -- Corresponds to IPP "multiple-
3032                                   -- document-handling"='separate-
3033                                   -- documents-collated-copies'
3034           uncollatedDocuments(5)  -- internal collated sheets,
3035                                   -- documents: A, A, ..., B, B, ...
3036                                   -- Corresponds to IPP "multiple-
3037                                   -- document-handling"='separate-
3038                                   -- documents-uncollated-copies'
3039       }
3040
3041
3042   JmJobSubmissionIDTypeTC ::= TEXTUAL-CONVENTION
3043       STATUS      current
3044       DESCRIPTION
3045           "Identifies the format type of a job submission ID.
3046
3047           Each job submission ID is a fixed-length, 48-octet printable
3048           US-ASCII [US-ASCII] coded character string containing no
3049           control characters, consisting of the fields defined in section
3050           3.5.1.
3051
3052           This is like a type 2 enumeration.  See section 3.7.3."
3053       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

```
3157
3158          canceled(7),
3159              A client has canceled the job and the server or device has
3160              completed canceling the job AND all MIB objects and
3161              attributes have reached their final values for the job.
3162              While the server or device is canceling the job, the job's
3163              jmJobStateReasons1 object SHOULD contain the
3164              processingToStopPoint value and one of the canceledByUser,
3165              canceledByOperator, or canceledAtDevice values.  The
3166              canceledByUser, canceledByOperator, or canceledAtDevice
3167              values remain while the job is in the canceled state.
3168
3169          aborted(8),
3170              The job has been aborted by the system, usually while the
3171              job was in the processing or processingStopped state and
3172              the server or device has completed aborting the job AND all
3173              MIB objects and attributes have reached their final values
3174              for the job.  While the server or device is aborting the
3175              job, the job's jmJobStateReasons1 object MAY contain the
3176              processingToStopPoint and abortedBySystem values.  If
3177              implemented, the abortedBySystem value SHALL remain while
3178              the job is in the aborted state.
3179
3180          completed(9)
3181              The job has completed successfully or with warnings or
3182              errors after processing and all of the media have been
3183              successfully stacked in the appropriate output bin(s) AND
3184              all MIB objects and attributes have reached their final
3185              values for the job.  The job's jmJobStateReasons1 object
3186              SHOULD contain one of: completedSuccessfully,
3187              completedWithWarnings, or completedWithErrors values.
3188
3189          This is a type 2 enumeration.  See Section 3.7.1.2."
3190      SYNTAX      INTEGER {
3191          unknown(2),
3192          pending(3),
3193          pendingHeld(4),
3194          processing(5),
3195          processingStopped(6),
3196          canceled(7),
3197          aborted(8),
3198          completed(9)
3199      }
```

```
3200
3201
3202   JmAttributeTypeTC ::= TEXTUAL-CONVENTION
3203       STATUS      current
3204       DESCRIPTION
3205           "The type of the attribute which identifies the attribute.
3206
3207           NOTE - The enum assignments are grouped logically with values
3208           assigned in groups of 20, so that additional values may be
3209           registered in the future and assigned a value that is part of
3210           their logical grouping.
3211
3212           Values in the range 2**30 to 2**31-1 are reserved for private
3213           or experimental usage.  This range corresponds to the same
3214           range reserved in IPP.  Implementers are warned that use of
3215           such values may conflict with other implementations.
3216           Implementers are encouraged to request registration of enum
3217           values following the procedures in Section 3.7.1.
3218
3219           See Section 3.2 entitled 'The Attribute Mechanism' for a
3220           description of this textual-convention and its use in the
3221           jmAttributeTable.  See Section 3.3.8 for the specification of
3222           each attribute.  The comment(s) after each enum assignment
3223           specifies the data type(s) of the attribute.
3224
3225           This is a type 2 enumeration.  See Section 3.7.1.2."
3226
3227       SYNTAX      INTEGER {
3228           other(1),                          -- Integer32 (-2..2147483647)
3229                                              -- AND/OR
3230                                              -- OCTET STRING(SIZE(0..63))
3231
3232           -- Job State attributes:
3233           jobStateReasons2(3),               -- JmJobStateReasons2TC
3234           jobStateReasons3(4),               -- JmJobStateReasons3TC
3235           jobStateReasons4(5),               -- JmJobStateReasons4TC
3236           processingMessage(6),              -- JmUTF8StringTC (SIZE(0..63))
3237           processingMessageNaturalLangTag(7),
3238                                              -- OCTET STRING(SIZE(0..63))
3239           jobCodedCharSet(8),                -- CodedCharSet
3240           jobNaturalLanguageTag(9),          -- OCTET STRING(SIZE(0..63))
3241
```

```
3242          -- Job Identification attributes:
3243          jobURI(20),                       -- OCTET STRING(SIZE(0..63))
3244          jobAccountName(21),               -- OCTET STRING(SIZE(0..63))
3245          serverAssignedJobName(22),        -- JmJobStringTC (SIZE(0..63))
3246          jobName(23),                      -- JmJobStringTC (SIZE(0..63))
3247          jobServiceTypes(24),              -- JmJobServiceTypesTC
3248          jobSourceChannelIndex(25),        -- Integer32 (0..2147483647)
3249          jobSourcePlatformType(26),        -- JmJobSourcePlatformTypeTC
3250          submittingServerName(27),         -- JmJobStringTC (SIZE(0..63))
3251          submittingApplicationName(28),    -- JmJobStringTC (SIZE(0..63))
3252          jobOriginatingHost(29),           -- JmJobStringTC (SIZE(0..63))
3253          deviceNameRequested(30),          -- JmJobStringTC (SIZE(0..63))
3254          queueNameRequested(31),           -- JmJobStringTC (SIZE(0..63))
3255          physicalDevice(32),               -- hrDeviceIndex
3256                                            -- AND/OR
3257                                            -- JmUTF8StringTC (SIZE(0..63))
3258          numberOfDocuments(33),            -- Integer32 (-2..2147483647)
3259          fileName(34),                     -- JmJobStringTC (SIZE(0..63))
3260          documentName(35),                 -- JmJobStringTC (SIZE(0..63))
3261          jobComment(36),                   -- JmJobStringTC (SIZE(0..63))
3262          documentFormatIndex(37),          -- Integer32 (0..2147483647)
3263          documentFormat(38),               -- PrtInterpreterLangFamilyTC
3264                                            -- AND/OR
3265                                            -- OCTET STRING(SIZE(0..63))
3266
3267          -- Job Parameter attributes:
3268          jobPriority(50),                  -- Integer32 (-2..100)
3269          jobProcessAfterDateAndTime(51), -- DateAndTime (SNMPv2-TC)
3270          jobHold(52),                      -- JmBooleanTC
3271          jobHoldUntil(53),                 -- JmJobStringTC (SIZE(0..63))
3272          outputBin(54),                    -- Integer32 (0..2147483647)
3273                                            -- AND/OR
3274                                            -- JmJobStringTC (SIZE(0..63))
3275          sides(55),                        -- Integer32 (-2..2)
3276          finishing(56),                    -- JmFinishingTC
3277
3278          -- Image Quality attributes:
3279          printQualityRequested(70),        -- JmPrintQualityTC
3280          printQualityUsed(71),             -- JmPrintQualityTC
3281          printerResolutionRequested(72), -- JmPrinterResolutionTC
3282          printerResolutionUsed(73),        -- JmPrinterResolutionTC
3283          tonerEcomonyRequested(74),        -- JmTonerEconomyTC
3284          tonerEcomonyUsed(75),             -- JmTonerEconomyTC
3285          tonerDensityRequested(76),        -- Integer32 (-2..100)
3286          tonerDensityUsed(77),             -- Integer32 (-2..100)
3287
```

```
3288          -- Job Progress attributes:
3289          jobCopiesRequested(90),          -- Integer32 (-2..2147483647)
3290          jobCopiesCompleted(91),          -- Integer32 (-2..2147483647)
3291          documentCopiesRequested(92),     -- Integer32 (-2..2147483647)
3292          documentCopiesCompleted(93),     -- Integer32 (-2..2147483647)
3293          jobKOctetsTransferred(94),       -- Integer32 (-2..2147483647)
3294          sheetCompletedCopyNumber(95),    -- Integer32 (-2..2147483647)
3295          sheetCompletedDocumentNumber(96),
3296                                           -- Integer32 (-2..2147483647)
3297          jobCollationType(97),            -- JmJobCollationTypeTC
3298
3299          -- Impression attributes:
3300          impressionsSpooled(110),         -- Integer32 (-2..2147483647)
3301          impressionsSentToDevice(111),    -- Integer32 (-2..2147483647)
3302          impressionsInterpreted(112),     -- Integer32 (-2..2147483647)
3303          impressionsCompletedCurrentCopy(113),
3304                                           -- Integer32 (-2..2147483647)
3305          fullColorImpressionsCompleted(114),
3306                                           -- Integer32 (-2..2147483647)
3307          highlightColorImpressionsCompleted(115),
3308                                           -- Integer32 (-2..2147483647)
3309
3310          -- Page attributes:
3311          pagesRequested(130),             -- Integer32 (-2..2147483647)
3312          pagesCompleted(131),             -- Integer32 (-2..2147483647)
3313          pagesCompletedCurrentCopy(132),  -- Integer32 (-2..2147483647)
3314
3315          -- Sheet attributes:
3316          sheetsRequested(150),            -- Integer32 (-2..2147483647)
3317          sheetsCompleted(151),            -- Integer32 (-2..2147483647)
3318          sheetsCompletedCurrentCopy(152),-- Integer32 (-2..2147483647)
3319
3320          -- Resource attributes:
3321          mediumRequested(170),            -- JmMediumTypeTC
3322                                           -- AND/OR
3323                                           -- JmJobStringTC (SIZE(0..63))
3324          mediumConsumed(171),             -- Integer32 (-2..2147483647)
3325                                           -- AND
3326                                           -- JmJobStringTC (SIZE(0..63))
3327          colorantRequested(172),          -- Integer32 (-2..2147483647)
3328                                           -- AND/OR
3329                                           -- JmJobStringTC (SIZE(0..63))
3330          colorantConsumed(173),           -- Integer32 (-2..2147483647)
3331                                           -- AND/OR
3332                                           -- JmJobStringTC (SIZE(0..63))
3333          mediumTypeConsumed(174),         -- Integer32 (-2..2147483647)
3334                                           -- AND
3335                                           -- JmJobStringTC (SIZE(0..63))
3336          mediumSizeConsumed(175),         -- Integer32 (-2..2147483647)
3337                                           -- AND
3338                                           -- JmJobStringTC (SIZE(0..63))
3339
```

```
3340          -- Time attributes:
3341          jobSubmissionToServerTime(190), -- JmTimeStampTC
3342                                          -- AND/OR
3343                                          -- DateAndTime
3344          jobSubmissionTime(191),         -- JmTimeStampTC
3345                                          -- AND/OR
3346                                          -- DateAndTime
3347          jobStartedBeingHeldTime(192),   -- JmTimeStampTC
3348                                          -- AND/OR
3349                                          -- DateAndTime
3350          jobStartedProcessingTime(193),  -- JmTimeStampTC
3351                                          -- AND/OR
3352                                          -- DateAndTime
3353          jobCompletionTime(194),         -- JmTimeStampTC
3354                                          -- AND/OR
3355                                          -- DateAndTime
3356          jobProcessingCPUTime(195)       -- Integer32 (-2..2147483647)
3357      }
```

```
3358
3359
3360   JmJobServiceTypesTC ::= TEXTUAL-CONVENTION
3361       STATUS      current
3362       DESCRIPTION
3363           "Specifies the type(s) of service to which the job has been
3364           submitted (print, fax, scan, etc.).  The service type is
3365           represented as an enum that is bit encoded with each job
3366           service type so that more general and arbitrary services can be
3367           created, such as services with more than one destination type,
3368           or ones with only a source or only a destination.  For example,
3369           a job service might scan, faxOut, and print a single job.  In
3370           this case, three bits would be set in the jobServiceTypes
3371           attribute, corresponding to the hexadecimal values: 0x8 + 0x20
3372           + 0x4, respectively, yielding: 0x2C.
3373
3374           Whether this attribute is set from a job attribute supplied by
3375           the job submission client or is set by the recipient job
3376           submission server or device depends on the job submission
3377           protocol.  With either implementation, the agent SHALL return a
3378           non-zero value for this attribute indicating the type of the
3379           job.
3380
3381           One of the purposes of this attribute is to permit a requester
3382           to filter out jobs that are not of interest.  For example, a
3383           printer operator MAY only be interested in jobs that include
3384           printing.  That is why the attribute is in the job
3385           identification category.
3386
3387           The following service component types are defined (in
3388           hexadecimal) and are assigned a separate bit value for use with
3389           the jobServiceTypes attribute:
3390
3391           other                                0x1
3392               The job contains some instructions that are not one of the
3393               identified types.
3394
3395           unknown                              0x2
3396               The job contains some instructions whose type is unknown to
3397               the agent.
3398
3399           print                                0x4
3400               The job contains some instructions that specify printing
3401
3402           scan                                 0x8
3403               The job contains some instructions that specify scanning
3404
```

```
3405            faxIn                               0x10
3406                The job contains some instructions that specify receive fax
3407
3408            faxOut                              0x20
3409                The job contains some instructions that specify sending fax
3410
3411            getFile                             0x40
3412                The job contains some instructions that specify accessing
3413                files or documents
3414
3415            putFile                             0x80
3416                The job contains some instructions that specify storing
3417                files or documents
3418
3419            mailList                            0x100
3420                The job contains some instructions that specify
3421                distribution of documents using an electronic mail system.
3422
3423            These bit definitions are the equivalent of a type 2 enum
3424            except that combinations of them MAY be used together.  See
3425            section 3.7.1.2."
3426        SYNTAX      INTEGER (0..2147483647)   -- 31 bits, all but sign bit
3427
3428
3429
3430    JmJobStateReasons1TC ::= TEXTUAL-CONVENTION
3431        STATUS      current
3432        DESCRIPTION
3433            "The JmJobStateReasonsNTC (N=1..4) textual-conventions are used
3434            with the jmJobStateReasons1 object and jobStateReasonsN
3435            (N=2..4), respectively, to provide additional information
3436            regarding the current jmJobState object value.  These values
3437            MAY be used with any job state or states for which the reason
3438            makes sense.  See section 3.3.9.1 for the specification of each
3439            bit value defined for use with the JmJobStateReasons1TC.
3440
3441            These bit definitions are the equivalent of a type 2 enum
3442            except that combinations of bits may be used together.  See
3443            section 3.7.1.2."
3444        SYNTAX      INTEGER (0..2147483647)   -- 31 bits, all but sign bit
3445
3446
3447
3448    JmJobStateReasons2TC ::= TEXTUAL-CONVENTION
3449        STATUS      current
3450        DESCRIPTION
3451            "This textual-convention is used with the jobStateReasons2
3452            attribute to provides additional information regarding the
3453            jmJobState object.  See section 3.3.9.2 for the specification
3454            of JmJobStateReasons2TC.  See section 3.3.9.1 for the
3455            description under JmJobStateReasons1TC for additional
3456            information that applies to all reasons.
```

```
3457
3458            These bit definitions are the equivalent of a type 2 enum
3459            except that combinations of them may be used together.  See
3460            section 3.7.1.2."
3461        SYNTAX      INTEGER (0..2147483647)   -- 31 bits, all but sign bit
3462
3463    JmJobStateReasons3TC ::= TEXTUAL-CONVENTION
3464        STATUS      current
3465        DESCRIPTION
3466            "This textual-convention is used with the jobStateReasons3
3467            attribute to provides additional information regarding the
3468            jmJobState object.  See section 3.3.9.3 for the specification
3469            of JmJobStateReasons3TC.  See section 3.3.9.1 for the
3470            description under JmJobStateReasons1TC for additional
3471            information that applies to all reasons.
3472
3473            These bit definitions are the equivalent of a type 2 enum
3474            except that combinations of them may be used together.  See
3475            section 3.7.1.2.  "
3476        SYNTAX      INTEGER (0..2147483647)   -- 31 bits, all but sign bit
3477
3478
3479
3480
3481
3482    JmJobStateReasons4TC ::= TEXTUAL-CONVENTION
3483        STATUS      current
3484        DESCRIPTION
3485            "This textual-convention is used in the jobStateReasons4
3486            attribute to provides additional information regarding the
3487            jmJobState object.  See section 3.3.9.4 for the specification
3488            of JmJobStateReasons4TC.  See section 3.3.9.1 for the
3489            description under JmJobStateReasons1TC for additional
3490            information that applies to all reasons.
3491
3492            These bit definitions are the equivalent of a type 2 enum
3493            except that combinations of them may be used together.  See
3494            section 3.7.1.2."
3495        SYNTAX      INTEGER (0..2147483647)   -- 31 bits, all but sign bit
```

```
3496
3497
3498  jobmonMIBObjects  OBJECT IDENTIFIER  ::= { jobmonMIB 1 }
3499
3500  -- The General Group (MANDATORY)
3501
3502  -- The jmGeneralGroup consists entirely of the jmGeneralTable.
3503
3504  jmGeneral  OBJECT IDENTIFIER ::= { jobmonMIBObjects 1 }
3505
3506  jmGeneralTable  OBJECT-TYPE
3507      SYNTAX      SEQUENCE OF JmGeneralEntry
3508      MAX-ACCESS  not-accessible
3509      STATUS      current
3510      DESCRIPTION
3511          "The jmGeneralTable consists of information of a general nature
3512          that are per-job-set, but are not per-job.  See Section 2
3513          entitled 'Terminology and Job Model' for the definition of a
3514          job set.
3515
3516          The MANDATORY-GROUP macro specifies that this group is
3517          MANDATORY."
3518      ::= { jmGeneral 1 }
3519
3520
3521  jmGeneralEntry  OBJECT-TYPE
3522      SYNTAX      JmGeneralEntry
3523      MAX-ACCESS  not-accessible
3524      STATUS      current
3525      DESCRIPTION
3526          "Information about a job set (queue).
3527
3528          An entry SHALL exist in this table for each job set."
3529      INDEX  { jmGeneralJobSetIndex }
3530      ::= { jmGeneralTable 1 }
3531
3532
3533  JmGeneralEntry ::= SEQUENCE {
3534      jmGeneralJobSetIndex                Integer32 (1..32767),
3535      jmGeneralNumberOfActiveJobs         Integer32 (0..2147483647),
3536      jmGeneralOldestActiveJobIndex       Integer32 (0..2147483647),
3537      jmGeneralNewestActiveJobIndex       Integer32 (0..2147483647),
3538      jmGeneralJobPersistence             Integer32 (15..2147483647),
3539      jmGeneralAttributePersistence       Integer32 (15..2147483647),
3540      jmGeneralJobSetName                 JmUTF8StringTC (SIZE(0..63))
3541  }
```

```
3542
3543  jmGeneralJobSetIndex OBJECT-TYPE
3544      SYNTAX      Integer32 (1..32767)
3545      MAX-ACCESS  not-accessible
3546      STATUS      current
3547      DESCRIPTION
3548          "A unique value for each job set in this MIB.  The jmJobTable
3549          and jmAttributeTable tables have this same index as their
3550          primary index.
3551
3552          The value(s) of the jmGeneralJobSetIndex SHALL be persistent
3553          across power cycles, so that clients that have retained
3554          jmGeneralJobSetIndex values will access the same job sets upon
3555          subsequent power-up.
3556
3557          An implementation that has only one job set, such as a printer
3558          with a single queue, SHALL hard code this object with the value
3559          1.
3560
3561          See Section 2 entitled 'Terminology and Job Model' for the
3562          definition of a job set.
3563          Corresponds to the first index in jmJobTable and
3564          jmAttributeTable."
3565      ::= { jmGeneralEntry 1 }
3566
3567
3568  jmGeneralNumberOfActiveJobs OBJECT-TYPE
3569      SYNTAX      Integer32 (0..2147483647)
3570      MAX-ACCESS  read-only
3571      STATUS      current
3572      DESCRIPTION
3573          "The current number of 'active' jobs in the jmJobIDTable,
3574          jmJobTable, and jmAttributeTable, i.e., the total number of
3575          jobs that are in the pending, processing, or processingStopped
3576          states.  See the JmJobStateTC textual-convention for the exact
3577          specification of the semantics of the job states."
3578      DEFVAL      { 0 }      -- no jobs
3579      ::= { jmGeneralEntry 2 }
```

```
3580
3581  jmGeneralOldestActiveJobIndex  OBJECT-TYPE
3582      SYNTAX      Integer32 (0..2147483647)
3583      MAX-ACCESS  read-only
3584      STATUS      current
3585      DESCRIPTION
3586          "The jmJobIndex of the oldest job that is still in one of the
3587          'active' states (pending, processing, or processingStopped).
3588          In other words, the index of the 'active' job that has been in
3589          the job tables the longest.
3590
3591          If there are no active jobs, the agent SHALL set the value of
3592          this object to 0.
3593
3594          See Section 3.2 entitled 'The Job Tables and the Oldest Active
3595          and Newest Active Indexes' for a description of the usage of
3596          this object."
3597      DEFVAL      { 0 }      -- no active jobs
3598      ::= { jmGeneralEntry 3 }
3599
3600
3601
3602  jmGeneralNewestActiveJobIndex  OBJECT-TYPE
3603      SYNTAX      Integer32 (0..2147483647)
3604      MAX-ACCESS  read-only
3605      STATUS      current
3606      DESCRIPTION
3607          "The jmJobIndex of the newest job that is in one of the
3608          'active' states (pending, processing, or processingStopped).
3609          In other words, the index of the 'active' job that has been
3610          most recently added to the job tables.
3611
3612          When all jobs become 'inactive', i.e., enter the pendingHeld,
3613          completed, canceled, or aborted states, the agent SHALL set the
3614          value of this object to 0.
3615
3616          See Section 3.2 entitled 'The Job Tables and the Oldest Active
3617          and Newest Active Indexes' for a description of the usage of
3618          this object."
3619      DEFVAL      { 0 }      -- no active jobs
3620      ::= { jmGeneralEntry 4 }
```

```
3621
3622  jmGeneralJobPersistence OBJECT-TYPE
3623      SYNTAX      Integer32 (15..2147483647)
3624      UNITS       "seconds"
3625      MAX-ACCESS  read-only
3626      STATUS      current
3627      DESCRIPTION
3628          "The minimum time in seconds for this instance of the Job Set
3629          that an entry SHALL remain in the jmJobIDTable and jmJobTable
3630          after processing has completed, i.e., the minimum time in
3631          seconds starting when the job enters the completed, canceled,
3632          or aborted state.
3633
3634          Configuring this object is implementation-dependent.
3635
3636          This value SHALL be equal to or greater than the value of
3637          jmGeneralAttributePersistence.  This value SHOULD be at least
3638          60 which gives a monitoring or accounting application one
3639          minute in which to poll for job data."
3640      DEFVAL      { 60 }          -- one minute
3641      ::= { jmGeneralEntry 5 }
3642
3643
3644
3645  jmGeneralAttributePersistence OBJECT-TYPE
3646      SYNTAX      Integer32 (15..2147483647)
3647      UNITS       "seconds"
3648      MAX-ACCESS  read-only
3649      STATUS      current
3650      DESCRIPTION
3651          "The minimum time in seconds for this instance of the Job Set
3652          that an entry SHALL remain in the jmAttributeTable after
3653          processing has completed , i.e., the time in seconds starting
3654          when the job enters the completed, canceled, or aborted state.
3655
3656          Configuring this object is implementation-dependent.
3657
3658          This value SHOULD be at least 60 which gives a monitoring or
3659          accounting application one minute in which to poll for job
3660          data."
3661      DEFVAL      { 60 }          -- one minute
3662      ::= { jmGeneralEntry 6 }
```

```
3663
3664   jmGeneralJobSetName OBJECT-TYPE
3665       SYNTAX      JmUTF8StringTC (SIZE(0..63))
3666       MAX-ACCESS  read-only
3667       STATUS      current
3668       DESCRIPTION
3669           "The human readable name of this job set assigned by the system
3670           administrator (by means outside of this MIB).  Typically, this
3671           name SHOULD be the name of the job queue.  If a server or
3672           device has only a single job set, this object can be the
3673           administratively assigned name of the server or device itself.
3674           This name does not need to be unique, though each job set in a
3675           single Job Monitoring MIB SHOULD have distinct names.
3676
3677           NOTE - If the job set corresponds to a single printer and the
3678           Printer MIB is implemented, this value SHOULD be the same as
3679           the prtGeneralPrinterName object in the draft Printer MIB
3680           [print-mib-draft].  If the job set corresponds to an IPP
3681           Printer, this value SHOULD be the same as the IPP 'printer-
3682           name' Printer attribute.
3683
3684           NOTE - The purpose of this object is to help the user of the
3685           job monitoring application distinguish between several job sets
3686           in implementations that support more than one job set.
3687
3688           See the OBJECT compliance macro for the minimum maximum length
3689           required for conformance."
3690       DEFVAL      { ''H }      -- empty string
3691       ::= { jmGeneralEntry 7 }
3692
3693
3694
```

```
3695
3696
3697   -- The Job ID Group (MANDATORY)
3698
3699   -- The jmJobIDGroup consists entirely of the jmJobIDTable.
3700
3701   jmJobID  OBJECT IDENTIFIER ::= { jobmonMIBObjects 2 }
3702
3703   jmJobIDTable  OBJECT-TYPE
3704       SYNTAX      SEQUENCE OF JmJobIDEntry
3705       MAX-ACCESS  not-accessible
3706       STATUS      current
3707       DESCRIPTION
3708           "The jmJobIDTable provides a correspondence map (1) between the
3709           job submission ID that a client uses to refer to a job and (2)
3710           the jmGeneralJobSetIndex and jmJobIndex that the Job Monitoring
3711           MIB agent assigned to the job and that are used to access the
3712           job in all of the other tables in the MIB.  If a monitoring
3713           application already knows the jmGeneralJobSetIndex and the
3714           jmJobIndex of the job it is querying, that application NEED NOT
3715           use the jmJobIDTable.
3716
3717           The MANDATORY-GROUP macro specifies that this group is
3718           MANDATORY."
3719       ::= { jmJobID 1 }
3720
3721
3722
3723   jmJobIDEntry  OBJECT-TYPE
3724       SYNTAX      JmJobIDEntry
3725       MAX-ACCESS  not-accessible
3726       STATUS      current
3727       DESCRIPTION
3728           "The map from (1) the jmJobSubmissionID to (2) the
3729           jmGeneralJobSetIndex and jmJobIndex.
3730
3731           An entry SHALL exist in this table for each job currently known
3732           to the agent for all job sets and job states.  There MAY be
3733           more than one jmJobIDEntry that maps to a single job.  This
3734           many to one mapping can occur when more than one network entity
3735           along the job submission path supplies a job submission ID.
3736           See Section 3.5.  However, each job SHALL appear once and in
3737           one and only one job set."
3738       INDEX  { jmJobSubmissionID }
3739       ::= { jmJobIDTable 1 }
3740
3741   JmJobIDEntry ::= SEQUENCE {
3742       jmJobSubmissionID                   OCTET STRING(SIZE(48)),
3743       jmJobIDJobSetIndex                  Integer32 (0..32767),
3744       jmJobIDJobIndex                     Integer32 (0..2147483647)
3745   }
```

```
3746
3747  jmJobSubmissionID OBJECT-TYPE
3748      SYNTAX      OCTET STRING(SIZE(48))
3749      MAX-ACCESS  not-accessible
3750      STATUS      current
3751      DESCRIPTION
3752          "A quasi-unique 48-octet fixed-length string ID which
3753          identifies the job within a particular client-server
3754          environment.  There are multiple formats for the
3755          jmJobSubmissionID.  Each format SHALL be uniquely identified.
3756          See the JmJobSubmissionIDTypeTC textual convention.  Each
3757          format SHALL be registered using the procedures of a type 2
3758          enum.  See section 3.7.3 entitled: 'PWG Registration of Job
3759          Submission Id Formats'.
3760
3761          If the requester (client or server) does not supply a job
3762          submission ID in the job submission protocol, then the
3763          recipient (server or device) SHALL assign a job submission ID
3764          using any of the standard formats that have been reserved for
3765          agents and adding the final 8 octets to distinguish the ID from
3766          others submitted from the same requester.
3767
3768          The monitoring application, whether in the client or running
3769          separately, MAY use the job submission ID to help identify
3770          which jmJobIndex was assigned by the agent, i.e., in which row
3771          the job information is in the other tables.
3772
3773          NOTE - fixed-length is used so that a management application
3774          can use a shortened GetNext varbind (in SNMPv1 and SNMPv2) in
3775          order to get the next submission ID, disregarding the remainder
3776          of the ID in order to access jobs independent of the trailing
3777          identifier part, e.g., to get all jobs submitted by a
3778          particular jmJobOwner or submitted from a particular MAC
3779          address.
3780
3781          See the JmJobSubmissionIDTypeTC textual convention.
3782          See APPENDIX B - Support of Job Submission Protocols."
3783      ::= { jmJobIDEntry 1 }
```

```
3784
3785  jmJobIDJobSetIndex OBJECT-TYPE
3786      SYNTAX      Integer32 (0..32767)
3787      MAX-ACCESS  read-only
3788      STATUS      current
3789      DESCRIPTION
3790          "This object contains the value of the jmGeneralJobSetIndex for
3791          the job with the jmJobSubmissionID value, i.e., the job set
3792          index of the job set in which the job was placed when that
3793          server or device accepted the job.  This 16-bit value in
3794          combination with the jmJobIDJobIndex value permits the
3795          management application to access the other tables to obtain the
3796          job-specific objects for this job.
3797
3798          See jmGeneralJobSetIndex in the jmGeneralTable."
3799      DEFVAL      { 0 }      -- 0 indicates no job set index
3800      ::= { jmJobIDEntry 2 }
3801
3802
3803
3804  jmJobIDJobIndex OBJECT-TYPE
3805      SYNTAX      Integer32 (0..2147483647)
3806      MAX-ACCESS  read-only
3807      STATUS      current
3808      DESCRIPTION
3809          "This object contains the value of the jmJobIndex for the job
3810          with the jmJobSubmissionID value, i.e., the job index for the
3811          job when the server or device accepted the job.  This value, in
3812          combination with the jmJobIDJobSetIndex value, permits the
3813          management application to access the other tables to obtain the
3814          job-specific objects for this job.
3815
3816          See jmJobIndex in the jmJobTable."
3817      DEFVAL      { 0 }      -- 0 indicates no jmJobIndex value.
3818      ::= { jmJobIDEntry 3 }
3819
3820
```

```
3821
3822
3823   -- The Job Group (MANDATORY)
3824
3825   -- The jmJobGroup consists entirely of the jmJobTable.
3826
3827   jmJob  OBJECT IDENTIFIER ::= { jobmonMIBObjects 3 }
3828
3829   jmJobTable  OBJECT-TYPE
3830       SYNTAX      SEQUENCE OF JmJobEntry
3831       MAX-ACCESS  not-accessible
3832       STATUS      current
3833       DESCRIPTION
3834           "The jmJobTable consists of basic job state and status
3835           information for each job in a job set that (1) monitoring
3836           applications need to be able to access in a single SNMP Get
3837           operation, (2) that have a single value per job, and (3) that
3838           SHALL always be implemented.
3839
3840           The MANDATORY-GROUP macro specifies that this group is
3841           MANDATORY."
3842       ::= { jmJob 1 }
3843
3844
3845
3846   jmJobEntry  OBJECT-TYPE
3847       SYNTAX      JmJobEntry
3848       MAX-ACCESS  not-accessible
3849       STATUS      current
3850       DESCRIPTION
3851           "Basic per-job state and status information.
3852
3853           An entry SHALL exist in this table for each job, no matter what
3854           the state of the job is.  Each job SHALL appear in one and only
3855           one job set.
3856
3857           See Section 3.2 entitled 'The Job Tables'."
3858       INDEX  { jmGeneralJobSetIndex, jmJobIndex }
3859       ::= { jmJobTable 1 }
3860
3861   JmJobEntry ::= SEQUENCE {
3862       jmJobIndex                          Integer32 (1..2147483647),
3863       jmJobState                          JmJobStateTC,
3864       jmJobStateReasons1                  JmJobStateReasons1TC,
3865       jmNumberOfInterveningJobs           Integer32 (-2..2147483647),
3866       jmJobKOctetsPerCopyRequested        Integer32 (-2..2147483647),
3867       jmJobKOctetsProcessed               Integer32 (-2..2147483647),
3868       jmJobImpressionsPerCopyRequested    Integer32 (-2..2147483647),
3869       jmJobImpressionsCompleted           Integer32 (-2..2147483647),
3870       jmJobOwner                          JmJobStringTC (SIZE(0..63))
3871   }
```

```
3872
3873  jmJobIndex OBJECT-TYPE
3874      SYNTAX      Integer32 (1..2147483647)
3875      MAX-ACCESS  not-accessible
3876      STATUS      current
3877      DESCRIPTION
3878          "The sequential, monatonically increasing identifier index for
3879          the job generated by the server or device when that server or
3880          device accepted the job.  This index value permits the
3881          management application to access the other tables to obtain the
3882          job-specific row entries.
3883
3884          See Section 3.2 entitled 'The Job Tables and the Oldest Active
3885          and Newest Active Indexes'.
3886          See Section 3.5 entitled 'Job Identification'.
3887          See also jmGeneralNewestActiveJobIndex for the largest value of
3888          jmJobIndex.
3889          See JmJobSubmissionIDTypeTC for a limit on the size of this
3890          index if the agent represents it as an 8-digit decimal number."
3891      ::= { jmJobEntry 1 }
3892
3893
3894
3895  jmJobState OBJECT-TYPE
3896      SYNTAX      JmJobStateTC
3897      MAX-ACCESS  read-only
3898      STATUS      current
3899      DESCRIPTION
3900          "The current state of the job (pending, processing, completed,
3901          etc.).  Agents SHALL implement only those states which are
3902          appropriate for the particular implementation.  However,
3903          management applications SHALL be prepared to receive all the
3904          standard job states.
3905
3906          The final value for this object SHALL be one of: completed,
3907          canceled, or aborted.  The minimum length of time that the
3908          agent SHALL maintain MIB data for a job in the completed,
3909          canceled, or aborted state before removing the job data from
3910          the jmJobIDTable and jmJobTable is specified by the value of
3911          the jmGeneralJobPersistence object."
3912      DEFVAL      { unknown }      -- default is unknown
3913      ::= { jmJobEntry 2 }
```

```
3914
3915  jmJobStateReasons1 OBJECT-TYPE
3916      SYNTAX      JmJobStateReasons1TC
3917      MAX-ACCESS  read-only
3918      STATUS      current
3919      DESCRIPTION
3920          "Additional information about the job's current state, i.e.,
3921          information that augments the value of the job's jmJobState
3922          object.
3923
3924          Implementation of any reason values is OPTIONAL, but an agent
3925          SHOULD return any reason information available.  These values
3926          MAY be used with any job state or states for which the reason
3927          makes sense.  Since the Job State Reasons will be more dynamic
3928          than the Job State, it is recommended that a job monitoring
3929          application read this object every time jmJobState is read.
3930          When the agent cannot provide a reason for the current state of
3931          the job, the value of the jmJobStateReasons1 object and
3932          jobStateReasonsN attributes SHALL be 0.
3933
3934          The jobStateReasonsN (N=2..4) attributes provide further
3935          additional information about the job's current state."
3936      DEFVAL      { 0 }      -- no reasons
3937      ::= { jmJobEntry 3 }
3938
3939
3940
3941  jmNumberOfInterveningJobs OBJECT-TYPE
3942      SYNTAX      Integer32 (-2..2147483647)
3943      MAX-ACCESS  read-only
3944      STATUS      current
3945      DESCRIPTION
3946          "The number of jobs that are expected to complete processing
3947          before this job has completed processing according to the
3948          implementation's queuing algorithm, if no other jobs were to be
3949          submitted.  In other words, this value is the job's queue
3950          position.  The agent SHALL return a value of 0 for this
3951          attribute when the job is the next job to complete processing
3952          (or has completed processing)."
3953      DEFVAL      { 0 }      -- default is no intervening jobs.
3954      ::= { jmJobEntry 4 }
```

```
3955
3956  jmJobKOctetsPerCopyRequested OBJECT-TYPE
3957      SYNTAX      Integer32 (-2..2147483647)
3958      MAX-ACCESS  read-only
3959      STATUS      current
3960      DESCRIPTION
3961          "The total size in K (1024) octets of the document(s) being
3962          requested to be processed in the job.  The agent SHALL round
3963          the actual number of octets up to the next highest K.  Thus 0
3964          octets is represented as '0', 1-1024 octets is represented as
3965          '1', 1025-2048 is represented as '2', etc.
3966
3967          In computing this value, the server/device SHALL NOT include
3968          the multiplicative factors contributed by (1) the number of
3969          document copies, and (2) the number of job copies, independent
3970          of whether the device can process multiple copies of the job or
3971          document without making multiple passes over the job or
3972          document data and independent of whether the output is collated
3973          or not.  Thus the server/device computation is independent of
3974          the implementation and indicates the size of the document(s)
3975          measured in K octets independent of the number of copies."
3976      DEFVAL      { -2 }      -- the default is unknown(-2)
3977      ::= { jmJobEntry 5 }
3978
3979
3980
3981  jmJobKOctetsProcessed OBJECT-TYPE
3982      SYNTAX      Integer32 (-2..2147483647)
3983      MAX-ACCESS  read-only
3984      STATUS      current
3985      DESCRIPTION
3986          "The total number of octets processed by the server or device
3987          measured in units of K (1024) octets so far.  The agent SHALL
3988          round the actual number of octets processed up to the next
3989          higher K.  Thus 0 octets is represented as '0', 1-1024 octets
3990          is represented as '1', 1025-2048 octets is '2', etc.  For
3991          printing devices, this value is the number interpreted by the
3992          page description language interpreter rather than what has been
3993          marked on media.
3994
3995          For implementations where multiple copies are produced by the
3996          interpreter with only a single pass over the data, the final
3997          value SHALL be equal to the value of the
3998          jmJobKOctetsPerCopyRequested object.  For implementations where
3999          multiple copies are produced by the interpreter by processing
4000          the data for each copy, the final value SHALL be a multiple of
4001          the value of the jmJobKOctetsPerCopyRequested object.
4002
4003          NOTE - See the impressionsCompletedCurrentCopy and
4004          pagesCompletedCurrentCopy attributes for attributes that are
4005          reset on each document copy.
4006
```

```
4007            NOTE - The jmJobKOctetsProcessed object can be used with the
4008            jmJobKOctetsPerCopyRequested object to provide an indication of
4009            the relative progress of the job, provided that the
4010            multiplicative factor is taken into account for some
4011            implementations of multiple copies."
4012        DEFVAL      { 0 }      -- default is no octets processed.
4013        ::= { jmJobEntry 6 }
4014
4015
4016    jmJobImpressionsPerCopyRequested OBJECT-TYPE
4017        SYNTAX      Integer32 (-2..2147483647)
4018        MAX-ACCESS  read-only
4019        STATUS      current
4020        DESCRIPTION
4021            "The total size in number of impressions of the document(s)
4022            submitted.
4023
4024            In computing this value, the server/device SHALL NOT include
4025            the multiplicative factors contributed by (1) the number of
4026            document copies, and (2) the number of job copies, independent
4027            of whether the device can process multiple copies of the job or
4028            document without making multiple passes over the job or
4029            document data and independent of whether the output is collated
4030            or not.  Thus the server/device computation is independent of
4031            the implementation and reflects the size of the document(s)
4032            measured in impressions independent of the number of copies.
4033
4034            See the definition of the term 'impression' in Section 2."
4035        DEFVAL      { -2 }      -- default is unknown(-2)
4036        ::= { jmJobEntry 7 }
4037
4038
4039    jmJobImpressionsCompleted OBJECT-TYPE
4040        SYNTAX      Integer32 (-2..2147483647)
4041        MAX-ACCESS  read-only
4042        STATUS      current
4043        DESCRIPTION
4044            "The total number of impressions completed for this job so far.
4045            For printing devices, the impressions completed includes
4046            interpreting, marking, and stacking the output.  For other
4047            types of job services, the number of impressions completed
4048            includes the number of impressions processed.
4049
4050            NOTE - See the impressionsCompletedCurrentCopy and
4051            pagesCompletedCurrentCopy attributes for attributes that are
4052            reset on each document copy.
4053
4054            NOTE - The jmJobImpressionsCompleted object can be used with
4055            the jmJobImpressionsPerCopyRequested object to provide an
4056            indication of the relative progress of the job, provided that
4057            the multiplicative factor is taken into account for some
4058            implementations of multiple copies.
```

```
4059
4060            See the definition of the term 'impression' in Section 2 and
4061            the counting example in Section 3.4 entitled 'Monitoring Job
4062            Progress'."
4063        DEFVAL      { 0 }      -- default is no octets
4064        ::= { jmJobEntry 8 }
4065
4066
4067
4068    jmJobOwner OBJECT-TYPE
4069        SYNTAX      JmJobStringTC (SIZE(0..63))
4070        MAX-ACCESS  read-only
4071        STATUS      current
4072        DESCRIPTION
4073            "The coded character set name of the user that submitted the
4074            job.  The method of assigning this user name will be system
4075            and/or site specific but the method MUST ensure that the name
4076            is unique to the network that is visible to the client and
4077            target device.
4078
4079            This value SHOULD be the most authenticated name of the user
4080            submitting the job.
4081
4082            See the OBJECT compliance macro for the minimum maximum length
4083            required for conformance."
4084        DEFVAL      { ''H }      -- default is empty string
4085        ::= { jmJobEntry 9 }
4086
4087
```

```
4088
4089
4090   -- The Attribute Group (MANDATORY)
4091
4092   -- The jmAttributeGroup consists entirely of the jmAttributeTable.
4093   --
4094   -- Implementation of the objects in this group is MANDATORY.
4095   -- See Section 3.1 entitled 'Conformance Considerations'.
4096   -- An agent SHALL implement any attribute if (1) the server or device
4097   -- supports the functionality represented by the attribute and (2) the
4098   -- information is available to the agent.
4099
4100   jmAttribute  OBJECT IDENTIFIER ::= { jobmonMIBObjects 4 }
4101
4102
4103
4104   jmAttributeTable  OBJECT-TYPE
4105       SYNTAX      SEQUENCE OF JmAttributeEntry
4106       MAX-ACCESS  not-accessible
4107       STATUS      current
4108       DESCRIPTION
4109           "The jmAttributeTable SHALL contain attributes of the job and
4110           document(s) for each job in a job set.  Instead of allocating
4111           distinct objects for each attribute, each attribute is
4112           represented as a separate row in the jmAttributeTable.
4113
4114           The MANDATORY-GROUP macro specifies that this group is
4115           MANDATORY.  An agent SHALL implement any attribute if (1) the
4116           server or device supports the functionality represented by the
4117           attribute and (2) the information is available to the agent. "
4118       ::= { jmAttribute 1 }
4119
4120
4121
```

```
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))
}
```

```
4172
4173  jmAttributeTypeIndex OBJECT-TYPE
4174      SYNTAX      JmAttributeTypeTC
4175      MAX-ACCESS  not-accessible
4176      STATUS      current
4177      DESCRIPTION
4178          "The type of attribute that this row entry represents.
4179
4180          The type MAY identify information about the job or document(s)
4181          or MAY identify a resource required to process the job before
4182          the job start processing and/or consumed by the job as the job
4183          is processed.
4184
4185          Examples of job attributes (i.e., apply to the job as a whole)
4186          that have only one instance per job include:
4187          jobCopiesRequested(90), documentCopiesRequested(92),
4188          jobCopiesCompleted(91), documentCopiesCompleted(93), while
4189          examples of job attributes that may have more than one instance
4190          per job include:  documentFormatIndex(37), and
4191          documentFormat(38).
4192
4193          Examples of document attributes (one instance per document)
4194          include: fileName(34), and documentName(35).
4195
4196          Examples of required and consumed resource attributes include:
4197          pagesRequested(130), mediumRequested(170), pagesCompleted(131),
4198          and mediumConsumed(171), respectively."
4199      ::= { jmAttributeEntry 1 }
4200
4201
4202
4203  jmAttributeInstanceIndex OBJECT-TYPE
4204      SYNTAX      Integer32 (1..32767)
4205      MAX-ACCESS  not-accessible
4206      STATUS      current
4207      DESCRIPTION
4208          "A running 16-bit index of the attributes of the same type for
4209          each job.  For those attributes with only a single instance per
4210          job, this index value SHALL be 1.  For those attributes that
4211          are a single value per document, the index value SHALL be the
4212          document number, starting with 1 for the first document in the
4213          job.  Jobs with only a single document SHALL use the index
4214          value of 1.  For those attributes that can have multiple values
4215          per job or per document, such as documentFormatIndex(37) or
4216          documentFormat(38), the index SHALL be a running index for the
4217          job as a whole, starting at 1."
4218      ::= { jmAttributeEntry 2 }
```

```
4219
4220  jmAttributeValueAsInteger OBJECT-TYPE
4221      SYNTAX      Integer32 (-2..2147483647)
4222      MAX-ACCESS  read-only
4223      STATUS      current
4224      DESCRIPTION
4225          "The integer value of the attribute.  The value of the
4226          attribute SHALL be represented as an integer if the enum
4227          description in the JmAttributeTypeTC textual-convention
4228          definition has the tag: 'INTEGER:'.
4229
4230          Depending on the enum definition, this object value MAY be an
4231          integer, a counter, an index, or an enum, depending on the
4232          jmAttributeTypeIndex value.  The units of this value are
4233          specified in the enum description.
4234
4235          For those attributes that are accumulating job consumption as
4236          the job is processed as specified in the JmAttributeTypeTC
4237          textual-convention, SHALL contain the final value after the job
4238          completes processing, i.e., this value SHALL indicate the total
4239          usage of this resource made by the job.
4240
4241          A monitoring application is able to copy this value to a
4242          suitable longer term storage for later processing as part of an
4243          accounting system.
4244
4245          Since the agent MAY add attributes representing resources to
4246          this table while the job is waiting to be processed or being
4247          processed, which can be a long time before any of the resources
4248          are actually used, the agent SHALL set the value of the
4249          jmAttributeValueAsInteger object to 0 for resources that the
4250          job has not yet consumed.
4251
4252          Attributes for which the concept of an integer value is
4253          meaningless, such as fileName(34), jobName, and
4254          processingMessage, do not have the 'INTEGER:' tag in the
4255          JmAttributeTypeTC definition and so an agent SHALL always
4256          return a value of '-1' to indicate 'other' for the value of the
4257          jmAttributeValueAsInteger object for these attributes.
4258
4259          For attributes which do have the 'INTEGER:' tag in the
4260          JmAttributeTypeTC definition, if the integer value is not (yet)
4261          known, the agent either (1) SHALL not materialize the row in
4262          the jmAttributeTable until the value is known or (2) SHALL
4263          return a '-2' to represent an 'unknown' counting integer value,
4264          a '0' to represent an 'unknown' index value, and a '2' to
4265          represent an 'unknown(2)' enum value."
4266      DEFVAL      { -2 }      -- default value is unknown(-2)
4267      ::= { jmAttributeEntry 3 }
```

```
4268
4269  jmAttributeValueAsOctets OBJECT-TYPE
4270      SYNTAX      OCTET STRING(SIZE(0..63))
4271      MAX-ACCESS  read-only
4272      STATUS      current
4273      DESCRIPTION
4274          "The octet string value of the attribute.  The value of the
4275          attribute SHALL be represented as an OCTET STRING if the enum
4276          description in the JmAttributeTypeTC textual-convention
4277          definition has the tag: 'OCTETS:'.
4278
4279          Depending on the enum definition, this object value MAY be a
4280          coded character set string (text), such as 'JmUTF8StringTC', or
4281          a binary octet string, such as 'DateAndTime'.
4282
4283          Attributes for which the concept of an octet string value is
4284          meaningless, such as pagesCompleted, do not have the tag
4285          'OCTETS:' in the JmAttributeTypeTC definition and so the agent
4286          SHALL always return a zero length string for the value of the
4287          jmAttributeValueAsOctets object.
4288
4289          For attributes which do have the 'OCTETS:' tag in the
4290          JmAttributeTypeTC definition, if the OCTET STRING value is not
4291          (yet) known, the agent either SHALL NOT materialize the row in
4292          the jmAttributeTable until the value is known or SHALL return a
4293          zero-length string."
4294      DEFVAL      { ''H }      -- empty string
4295      ::= { jmAttributeEntry 4 }
```

```
4296  -- Notifications and Trapping
4297  -- Reserved for the future
4298
4299  jobmonMIBNotifications  OBJECT IDENTIFIER  ::= { jobmonMIB 2 }
4300
4301
4302
4303  -- Conformance Information
4304
4305  jmMIBConformance OBJECT IDENTIFIER ::= { jobmonMIB 3 }
4306
4307
4308
4309  -- compliance statements
4310  jmMIBCompliance MODULE-COMPLIANCE
4311      STATUS  current
4312      DESCRIPTION
4313          "The compliance statement for agents that implement the
4314          job monitoring MIB."
4315      MODULE -- this module
4316      MANDATORY-GROUPS {
4317          jmGeneralGroup, jmJobIDGroup, jmJobGroup, jmAttributeGroup }
4318
4319      OBJECT   jmGeneralJobSetName
4320      SYNTAX   JmUTF8StringTC (SIZE(0..8))
4321      DESCRIPTION
4322          "Only 8 octets maximum string length NEED be supported by the
4323          agent."
4324
4325      OBJECT   jmJobOwner
4326      SYNTAX   JmJobStringTC (SIZE(0..16))
4327      DESCRIPTION
4328          "Only 16 octets maximum string length NEED be supported by the
4329          agent."
4330
4331  -- There are no CONDITIONALLY MANDATORY or OPTIONAL groups.
4332
4333      ::= { jmMIBConformance 1 }
4334
```

```
4335  jmMIBGroups      OBJECT IDENTIFIER ::= { jmMIBConformance 2 }
4336
4337  jmGeneralGroup OBJECT-GROUP
4338      OBJECTS {
4339          jmGeneralNumberOfActiveJobs,   jmGeneralOldestActiveJobIndex,
4340          jmGeneralNewestActiveJobIndex, jmGeneralJobPersistence,
4341          jmGeneralAttributePersistence, jmGeneralJobSetName}
4342      STATUS  current
4343      DESCRIPTION
4344          "The general group."
4345      ::= { jmMIBGroups 1 }
4346
4347
4348
4349  jmJobIDGroup OBJECT-GROUP
4350      OBJECTS {
4351          jmJobIDJobSetIndex, jmJobIDJobIndex }
4352      STATUS  current
4353      DESCRIPTION
4354          "The job ID group."
4355      ::= { jmMIBGroups 2 }
4356
4357
4358
4359  jmJobGroup OBJECT-GROUP
4360      OBJECTS {
4361          jmJobState, jmJobStateReasons1, jmNumberOfInterveningJobs,
4362          jmJobKOctetsPerCopyRequested, jmJobKOctetsProcessed,
4363          jmJobImpressionsPerCopyRequested, jmJobImpressionsCompleted,
4364          jmJobOwner }
4365      STATUS  current
4366      DESCRIPTION
4367          "The job group."
4368      ::= { jmMIBGroups 3 }
4369
4370
4371
4372  jmAttributeGroup OBJECT-GROUP
4373      OBJECTS {
4374          jmAttributeValueAsInteger, jmAttributeValueAsOctets }
4375      STATUS  current
4376      DESCRIPTION
4377          "The attribute group."
4378      ::= { jmMIBGroups 4 }
4379
4380
4381  END
```

5 Appendix A - Implementing the Job Life Cycle

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

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

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

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

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

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

6 APPENDIX B - Support of Job Submission Protocols

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

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

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

7 References

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[print-mib-draft] Turner, R., "Printer MIB", work in progress, on the standards track as a draft standard: <draft-ietf-printmib-mib-info-04.txt>, January 22, 1999.

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

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

[RFC1179] McLaughlin, L., III, "Line Printer Daemon Protocol", RFC 1179, August 1990

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

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

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

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

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

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

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

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

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

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

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

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

8 Notices

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.

El Segundo, CA 90245

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

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

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

Harry Lewis
IBM Corporation
6300 Diagonal Hwy
Boulder, CO 80301

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

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

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

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

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

http://www.pwg.org/

Other Participants:
Chuck Adams - Tektronix
Jeff Barnett - IBM
Keith Carter, IBM Corporation
Jeff Copeland - QMS
Andy Davidson - Tektronix
Roger deBry - IBM
Mabry Dozier - QMS
Lee Farrell - Canon

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

10 Change History

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

10.1Changes to produce version 1.0, dated February 19, 1999

The following changes were made to version 1.2, dated October 2, 1998 to make version 1.0 [sic], dated January 28, 1999:

1. Changed the version number back to 1.0 for this INTERNET-DRAFT in anticipation of its being published as an Information RFC.

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 as requested by the IESG.

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.

colorantConsumed, 42
colorantRequested, 41
deviceNameRequested, 31
documentCopiesCompleted, 36
documentCopiesRequested, 36
documentFormat, 33
documentFormatIndex, 32
documentName, 32
fileName, 32
finishing, 35
fullColorImpressionsCompleted, 38
highlightColorImpressionsCompleted, 39
impressionsCompletedCurrentCopy, 38
impressionsInterpreted, 38
impressionsSentToDevice, 38
impressionsSpooled, 38
jmAttributeInstanceIndex, 103
jmAttributeTypeIndex, 103
JmAttributeTypeTC, 80
jmAttributeValueAsInteger, 104
jmAttributeValueAsOctets, 105
JmBooleanTC, 74
JmFinishingTC, 72
jmGeneralAttributePersistence, 90
jmGeneralJobPersistence, 90
jmGeneralJobSetIndex, 88
jmGeneralJobSetName, 91
jmGeneralNewestActiveJobIndex, 89
jmGeneralNumberOfActiveJobs, 88
jmGeneralOldestActiveJobIndex, 89
JmJobCollationTypeTC, 76
jmJobIDJobIndex, 94
jmJobIDJobSetIndex, 94
jmJobImpressionsCompleted, 99
jmJobImpressionsPerCopyRequested, 99
jmJobIndex, 96
jmJobKOctetsPerCopyRequested, 98
jmJobKOctetsProcessed, 98
jmJobOwner, 100
JmJobServiceTypesTC, 84

JmJobSourcePlatformTypeTC, 71
jmJobState, 96
jmJobStateReasons1, 97
JmJobStateReasons1TC, 85
JmJobStateReasons2TC, 85
JmJobStateReasons3TC, 86
JmJobStateReasons4TC, 86
JmJobStateTC, 77
JmJobStringTC, 70
jmJobSubmissionID, 93
JmJobSubmissionIDTypeTC, 76
JmMediumTypeTC, 74
JmNaturalLanguageTagTC, 70
jmNumberOfInterveningJobs, 97
JmPrinterResolutionTC, 73
JmPrintQualityTC, 73
jmSystemAttrIntegerSupport, 105
JmTimeStampTC, 71
JmTonerEconomyTC, 74
JmUTF8StringTC, 70
jobAccountName, 28
jobCodedCharSet, 27
jobCollationType, 37
jobComment, 32
jobCompletionTime, 44
jobCopiesCompleted, 36
jobCopiesRequested, 36
jobHold, 34
jobHoldUntil, 34
jobKOctetsTransferred, 37
jobName, 29
jobNaturalLanguageTag, 27
jobOriginatingHost, 31
jobPriority, 33
jobProcessAfterDateAndTime, 34
jobProcessingCPUTime, 44
jobServiceTypes, 30
jobSourceChannelIndex, 30
jobSourcePlatformType, 30
jobStartedBeingHeldTime, 43
jobStartedProcessingTime, 44
jobStateReasons2, 25
jobStateReasons3, 25
jobStateReasons4, 25
jobSubmissionTime, 43
jobSubmissionToServerTime, 43
jobURI, 28
mediumConsumed, 41
mediumRequested, 41
mediumSizeConsumed, 42
mediumTypeConsumed, 42
numberOfDocuments, 31

other, 25
outputBin, 34
pagesCompleted, 39
pagesCompletedCurrentCopy, 40
pagesRequested, 39
physicalDevice, 31
printerResolutionRequested, 35
printerResolutionUsed, 35
printQualityRequested, 35
printQualityUsed, 35
processingMessage, 26
processingMessageNaturalLangTag, 26
queueNameRequested, 31
serverAssignedJobName, 28
sheetCompletedCopyNumber, 37
sheetCompletedDocumentNumber, 37
sheetsCompleted, 40
sheetsCompletedCurrentCopy, 40
sheetsRequested, 40
sides, 34
submittingApplicationName, 30
submittingServerName, 30
tonerDensityRequested, 35
tonerDensityUsed, 36
tonerEcomonyRequested, 35
tonerEcomonyUsed, 35