

1. Job Monitoring MIB, V0.84

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

From: Tom Hastings

Date: 07/21/97

Version: 0.84

File: ftp://ftp.pwg.org/pub/jmp/mibs/jmp-mib.doc .pdf jmp-mibr.doc .pdf .pdr

Status: Seventh draft MIB that corresponds to editorial comments on V0.83 and changes to keep in alignment with IPP (printer-resolution syntax). See the change history in the separate file: changes.doc .pdf.

We agreed that the MIB specification is finished except for any editorial comments that people may have. We resolved all PWG issues. I've included Ron Bergman's and David Perkin's extensive editorial comments. A small number of issues came from IETF reviewers (David Perkins and Ron Bergman), which have not been resolved. See the separate issues.doc and .pdf file.

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

The MIB has been greatly simplified so that now there are only 18 objects in the MIB. There are 65 attributes.

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

24 INTERNET-DRAFT
25
26
27
28
29
30
31
32
33

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

Job Monitoring MIB - V0.84

<draft-ietf-printmib-job-monitor-04.txt>

Expires Jan 21, 1997

34
35
36
37

Status of this Memo

39 This document is an Internet-Draft. Internet-Drafts are working documents of the
40 Internet Engineering Task Force (IETF), its areas, and its working groups. Note
41 that other groups may also distribute working documents as Internet-Drafts.
42 Internet-Drafts are draft documents valid for a maximum of six months and may be
43 updated, replaced, or obsoleted by other documents at any time. It is
44 inappropriate to use Internet-Drafts as reference material or to cite them other than
45 as "work in progress."
46 To learn the current status of any Internet-Draft, please check the "Iid-
47 abstracts.txt" listing contained in the Internet-Drafts Shadow Directories on
48 ftp.is.co.za (Africa), nic.nordu.net (Europe), munnari.oz.au (Pacific Rim),
49 ds.internic.net (US East Coast), or ftp.isi.edu (US West Coast).

Abstract

50
51 This Internet-Draft specifies a small set of read-only SNMP MIB objects for (1)
52 monitoring the status and progress of print jobs (2) obtaining resource
53 requirements before a job is processed, (3) monitoring resource consumption while
54 a job is being processed and (4) collecting resource accounting data after the
55 completion of a job. This MIB is intended to be implemented (1) in a printer or
56 (2) in a server that supports one or more printers. Use of the object set is not
57 limited to printing. However, support for services other than printing is outside
58 the scope of this Job Monitoring MIB. Future extensions to this MIB may include,
59 but are not limited to, fax machines and scanners.

60

61

TABLE OF CONTENTS

62 **1. INTRODUCTION..... 9**

63 **1.1 Types of Information in the MIB9**

64 **1.2 Types of Job Monitoring Applications10**

65 **2. TERMINOLOGY AND JOB MODEL 11**

66 **2.1 System Configurations for the Job Monitoring MIB14**

67 2.1.1 Configuration 1 - client-printer14

68 2.1.2 Configuration 2 - client-server-printer - agent in the server15

69 2.1.3 Configuration 3 - client-server-printer - client monitors printer agent and server16

70 **3. MANAGED OBJECT USAGE..... 17**

71 **3.1 Conformance Considerations.....17**

72 3.1.1 Conformance Terminology18

73 3.1.2 Agent Conformance Requirements18

74 3.1.2.1 MIB II System Group objects18

75 3.1.2.2 MIB II Interface Group objects18

76 3.1.2.3 Printer MIB objects19

77 3.1.3 Job Monitoring Application Conformance Requirements19

78 **3.2 The Job Tables and the Oldest Active and Newest Active Indexes19**

79 **3.3 The Attribute Mechanism.....21**

80 3.3.1 Conformance of Attribute Implementation.....22

81 3.3.2 Useful, 'Unknown', and 'Other' Values for Objects and Attributes23

82 3.3.3 Data Sub-types and Attribute Naming Conventions23

83 3.3.4 Single-Value (Row) Versus Multi-Value (MULTI-ROW) Attributes.....23

84 3.3.5 Requested Attributes.....24

85 3.3.6 Consumption Attributes.....24

86 3.3.7 Index Value Attributes24

87 **3.4 Job Identification24**

88 **3.5 Internationalization Considerations25**

89 **3.6 IANA Considerations.....25**

90 3.6.1 IANA Registration of enums26

91 3.6.1.1 Type 1 enumerations.....26

92 3.6.1.2 Type 2 enumerations.....26

93 3.6.1.3 Type 3 enumeration27

94 3.6.2 IANA Registration of type 2 bit values27

95 3.6.3 IANA Registration of Job Submission Id Formats.....27

96 3.6.4 IANA Registration of MIME types/sub-types for document-formats27

97 **3.7 Security Considerations.....27**

98 3.7.1 Read-Write objects27

99 3.7.2 Read-Only Objects In Other User's Jobs.....28

100 **3.8 Values for Objects.....22**

101 **3.9 Notification-s28**

102 **4. MIB SPECIFICATION 28**

103 **Textual conventions for this MIB module30**

104 JmTimeStampTC - simple time in seconds.....30

105 JmJobSourcePlatformTypeTC - operating system platform definitions.....30

106 JmFinishingTC - device finishing definitions.....31

107 JmPrintQualityTC - print quality.....32

108 JmPrinterResolutionTC - printer resolution.....33

109 JmTonerEconomyTC - toner economy setting.....33

110 JmBooleanTC - Boolean value34

111 JmMediumTypeTC - medium type definitions.....34

112 JmJobSubmissionIDTypeTC - job submission ID type definitions.....35

113 JmJobStateTC - job state definitions.....37

114 JmAttributeTypeTC - attribute type definitions.....40

115 other (Int32(-2..) and/or Octets63)40

116 Job State attributes.....40

117 jobStateReasons2 (JmJobStateReasons2TC)41

118 jobStateReasons3 (JmJobStateReasons3TC)41

119 jobStateReasons4 (JmJobStateReasons4TC)41

120 processingMessage (Octets63).....41

121 Job Identification attributes.....41

122 jobAccountName (Octets63).....41

123 serverAssignedJobName (Octets63)41

124 jobName (Octets63).....42

125 jobServiceTypes (JmJobServiceTypesTC).....42

126 jobSourceChannelIndex (Int32(0..)).....43

127 jobSourcePlatformType (JmJobSourcePlatformTypeTC).....43

128 submittingServerName (Octets63).....43

129 submittingApplicationName (Octets63)43

130 jobOriginatingHost (Octets63).....43

131 deviceNameRequested (Octets63).....43

132 queueNameRequested (Octets63)43

133 physicalDevice (hrDeviceIndex and/or Octets63)43

134 numberOfDocuments (Int32(-2..)).....44

135 fileName (Octets63).....44

136 documentName (Octets63).....44

137 jobComment (Octets63)44
 138 documentFormatIndex (Int32(0..))44
 139 documentFormat (PrtInterpreterLangFamilyTC and/or Octets63).....44
 140 Job Parameter attributes.....45
 141 jobPriority (Int32(1..100)).....45
 142 jobProcessAfterDateAndTime (DateAndTime).....45
 143 jobHold (JmBooleanTC).....45
 144 jobHoldUntil (Octets63)45
 145 outputBin (Int32(0..) and/or Octets63)45
 146 sides (Int32(-2..2))46
 147 finishing (JmFinishingTC).....46
 148 Image Quality attributes (requested and used).....46
 149 printQualityRequested (JmPrintQualityTC).....46
 150 printQualityUsed (JmPrintQualityTC).....46
 151 printerResolutionRequested (JmPrinterResolutionTC).....46
 152 printerResolutionUsed (JmPrinterResolutionTC).....46
 153 tonerEcomonyRequested (JmTonerEconomyTC).....46
 154 tonerEcomonyUsed (JmTonerEconomyTC).....46
 155 tonerDensityRequested (Int32(-2..100)).....46
 156 tonerDensityUsed (Int32(-2..100)).....47
 157 Job Progress attributes (requested and consumed)47
 158 jobCopiesRequested (Int32(-2..)).....47
 159 jobCopiesCompleted (Int32(-2..)).....47
 160 documentCopiesRequested (Int32(-2..)).....47
 161 documentCopiesCompleted (Int32(-2..)).....47
 162 jobKOctetsTransferred (Int32(-2..)).....47
 163 Impression attributes (requested and consumed)48
 164 impressionsSpooled (Int32(-2..)).....48
 165 impressionsSentToDevice (Int32(-2..)).....48
 166 impressionsInterpreted (Int32(-2..))48
 167 impressionsCompletedCurrentCopy (Int32(-2..)).....48
 168 fullColorImpressionsCompleted (Int32(-2..)).....48
 169 highlightColorImpressionsCompleted (Int32(-2..)).....48
 170 Page attributes (requested and consumed).....49
 171 pagesRequested (Int32(-2..))49
 172 pagesCompleted (Int32(-2..))49
 173 pagesCompletedCurrentCopy (Int32(-2..))49
 174 Sheet attributes (requested and consumed).....49
 175 sheetsRequested (Int32(-2..)).....49
 176 sheetsCompleted (Int32(-2..)).....49
 177 sheetsCompletedCurrentCopy (Int32(-2..)).....49
 178 Resource attributes (requested and consumed)50
 179 mediumRequested (JmMediumTypeTC and/or Octets63)50
 180 mediumConsumed (Octets63)50
 181 colorantRequested (Int32(-2..) and/or Octets63)50
 182 colorantConsumed (Int32(-2..) and/or Octets63)50
 183 Time attributes (set by server or device).....50
 184 jobSubmissionToServerTime (JmTimeStampTC and/or DateAndTime).....51
 185 jobSubmissionTime (JmTimeStampTC and/or DateAndTime).....51

186 jobStartedBeingHeldTime (JmTimeStampTC and/or DateAndTime)51

187 jobStartedProcessingTime (JmTimeStampTC and/or DateAndTime)51

188 jobCompletedTime (JmTimeStampTC and/or DateAndTime)52

189 jobProcessingCPUTime (Int32(-2..))52

190 JmJobServiceTypesTC - bit encoded job service type definitions.....54

191 JmJobStateReasons1TC - additional information about job states55

192 JmJobStateReasons2TC - More additional information about job states58

193 JmJobStateReasons3TC - More additional information about job states61

194 JmJobStateReasons4TC - More additional information about job states61

195 **The General Group (MANDATORY)63**

196 jmGeneralJobSetIndex (Int32(1..32767)).....63

197 jmGeneralNumberOfActiveJobs (Int32(0..)).....64

198 jmGeneralOldestActiveJobIndex (Int32(0..)).....64

199 jmGeneralNewestActiveJobIndex (Int32(0..)).....64

200 jmGeneralJobPersistence (Int32(15..)).....65

201 jmGeneralAttributePersistence (Int32(15..)).....65

202 jmGeneralJobSetName (Octets63).....65

203 **The Job ID Group (MANDATORY).....66**

204 jmJobSubmissionID (OCTET STRING(SIZE(48))).....67

205 jmJobIDJobSetIndex (Int32(1..32767)).....67

206 jmJobIDJobIndex (Int32(1..)).....67

207 **The Job Group (MANDATORY).....68**

208 jmJobIndex (Int32(1..)).....69

209 jmJobState (JmJobStateTC).....69

210 jmJobStateReasons1 (JmJobStateReasons1TC).....69

211 jmNumberOfInterveningJobs (Int32(-2..))70

212 jmJobKOctetsRequested (Int32(-2..)).....70

213 jmJobKOctetsProcessed (Int32(-2..))70

214 jmJobImpressionsRequested (Int32(-2..)).....71

215 jmJobImpressionsCompleted (Int32(-2..)).....71

216 jmJobOwner (Octets63).....71

217 **The Attribute Group (MANDATORY)72**

218 jmAttributeTypeIndex (JmAttributeTypeTC).....73

219 jmAttributeInstanceIndex (Int32(1..32767))73

220 jmAttributeValueAsInteger (Int32(-2..)).....74

221 jmAttributeValueAsOctets (Octets63)74

222 **5. APPENDIX A - IMPLEMENTING THE JOB LIFE CYCLE..... 78**

223 **6. APPENDIX B - SUPPORT OF THE JOB SUBMISSION ID IN JOB**

224 **SUBMISSION PROTOCOLS 78**

225 **6.1 Hewlett-Packard's Printer Job Language (PJJ)79**

226 **6.2 ISO DPA.....79**

227 **7. REFERENCES..... 79**

228 **8. AUTHOR'S ADDRESSES..... 80**

229 **9. INDEX 83**

230

231

Job Monitoring MIB

232 1. Introduction

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

242 The Job Monitoring MIB consists of a General Group, a Job Submission ID Group, a Job
243 Group, and an Attribute Group. Each group is a table. All accessible objects are read-
244 only. The General Group contains general information that applies to all jobs in a job set.
245 The Job Submission ID table maps the job submission ID that the client uses to identify a
246 job to the **jmJobIndex** that the Job Monitoring Agent uses to identify jobs in the Job and
247 Attribute tables. The Job table contains the MANDATORY integer job state and status
248 objects. The Attribute table consists of multiple entries per job that specify (1) job and
249 document identification and parameters, (2) requested resources, and (3) consumed
250 resources during and after job processing/printing. Sixty five job attributes are defined as
251 textual conventions that an agent SHALL return if the server or device implements the
252 functionality so represented and the agent has access to the information.

253 1.1 Types of Information in the MIB

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

256 User:

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

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

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

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

264 Operator:

265 Provide a presentation of the state of all the jobs in the print system.
266 Provide the ability to identify the user that submitted the print job.
267 Provide the ability to identify the resources required by each job.
268 Provide the ability to define which physical printers are candidates for the print
269 job.
270 Provide some idea of how long each job will take. However, exact estimates of
271 time to process a job is not being attempted. Instead, objects are included that
272 allow the operator to be able to make gross estimates.

273 Capacity Planner:

274 Provide the ability to determine printer utilization as a function of time.
275 Provide the ability to determine how long jobs wait before starting to print.

276 Accountant:

277 Provide information to allow the creation of a record of resources consumed and
278 printer usage data for charging users or groups for resources consumed.
279 Provide information to allow the prediction of consumable usage and resource
280 need.

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

286 **1.2 Types of Job Monitoring Applications**

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

- 288 1. Monitor a single job starting when the job is submitted and ending a defined period
289 after the job completes. The Job Submission ID table provides the map to find the
290 specific job to be monitored.
- 291 2. Monitor all 'active' jobs in a queue, which this specification generalizes to a "job
292 set". End users may use such a program when selecting a least busy printer, so the
293 MIB is designed for such a program to start up quickly and find the information
294 needed quickly without having to read all (completed) jobs in order to find the
295 active jobs. System operators may also use such a program, in which case it would
296 be running for a long period of time and may also be interested in the jobs that have
297 completed. Finally such a program may be used to provide an enhanced console
298 and logging capability.

299 3. Collect resource usage for accounting or system utilization purposes that copy the
300 completed job statistics to an accounting system. It is recognized that depending on
301 accounting programs to copy MIB data during the job-retention period is
302 somewhat unreliable, since the accounting program may not be running (or may
303 have crashed). Such a program is also expected to keep a shadow copy of the
304 entire Job **Attribute** table including **completed, canceled, and aborted** jobs which
305 the program updates on each polling cycle. Such a program polls at the rate of the
306 persistence of the **Attribute** table. The design is not optimized to help such an
307 application determine which jobs are **completed, canceled, or aborted**. Instead,
308 the application SHALL query each job that the application's shadow copy shows
309 was not **complete, canceled, or aborted** at the previous poll cycle to see if it is
310 now **complete** or **canceled**, plus any new jobs that have been submitted.

311 The MIB provides a set of objects that represent a compatible subset of job and document
312 attributes of the ISO DPA standard[iso-dpa] and the Internet Printing Protocol (IPP)[ipp-
313 model], so that coherence is maintained between these two protocols and the information
314 presented to end users and system operators by monitoring applications. However, the
315 job monitoring MIB is intended to be used with printers that implement other job
316 submitting and management protocols, such as IEEE 1284.1 (TIPSI)[tipsi], as well as
317 with ones that do implement ISO DPA. Thus the job monitoring MIB does not require
318 implementation of either the ISO DPA or IPP protocols.

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

321 2. Terminology and Job Model

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

324 NOTE - Existing systems use conflicting terms, so these terms are drawn from the ISO
325 10175 Document Printing Application (DPA) standard[iso-dpa]. For example,
326 PostScript systems use the term *session* for what is called a *job* in this specification and
327 the term *job* to mean what is called a *document* in this specification. PjL systems use
328 the term *job* to mean what is called a *job* in this specification. PjL also supports
329 multiple *documents* per job, but does not support specifying per-document attributes
330 independently for each document.

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

333 Job Set: a group of jobs that are queued and scheduled together according to a specified
334 scheduling algorithm for a specified device or set of devices. For implementations that
335 embed the SNMP agent in the device, the MIB job set normally represents *all* the jobs

336 known to the device, so that the implementation only implements a single job set. If the
337 SNMP agent is implemented in a server that controls one or more devices, each MIB job
338 set represents a job queue for (1) a specific device or (2) set of devices, if the server uses a
339 single queue to load balance between several devices. Each job set is disjoint; no job
340 SHALL be represented in more than one MIB job set.

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

343 Client: the network entity that *end users* use to submit jobs to *spoolers, servers, or*
344 *printers* and other *devices*, depending on the configuration, using any job submission
345 protocol.

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

349 Device: a hardware entity that (1) interfaces to humans in human perceptible means, such
350 as produces marks on paper, scans marks on paper to produce an electronic
351 representations, or writes CD-ROMs or (2) interfaces electronically to another device,
352 such as sends FAX data to another FAX device.

353 Printer: a *device* that puts marks on media.

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

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

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

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

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

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

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

- 372 Proxy: an agent that acts as a concentrator for one or more other agents by accepting
373 SNMP operations on the behalf of one or more other agents, forwarding them on to those
374 other agents, gathering responses from those other agents and returning them to the
375 original requesting monitor.
- 376 User: a person that uses a client or a monitor.
- 377 End User: a user that uses a client to submit a print job.
- 378 System Operator: a user that uses a monitor to monitor the system and carries out tasks
379 to keep the system running.
- 380 System Administrator: a user that specifies policy for the system.
- 381 Job Instruction: an instruction specifying how, when, or where the job is to be processed.
382 Job instructions MAY be passed in the job submission protocol or MAY be embedded in
383 the document data or a combination depending on the job submission protocol and
384 implementation.
- 385 Document Instruction: an instruction specifying how to process the document.
386 Document instructions MAY be passed in the job submission protocol separate from the
387 actual document data, or MAY be embedded in the document data or a combination,
388 depending on the job submission protocol and implementation.
- 389 SNMP Information Object: a name, value-pair that specifies an action, a status, or a
390 condition in an SNMP MIB. Objects are identified in SNMP by an OBJECT
391 IDENTIFIER.
- 392 Attribute: a name, value-pair that specifies a job or document instruction, a status, or a
393 condition of a job or a document that has been submitted to a server or device. A
394 particular attribute NEED NOT be present in each job instance. In other words, attributes
395 are present in a job instance only when there is a need to express the value, either because
396 (1) the client supplied a value in the job submission protocol, (2) the document data
397 contained an embedded attribute, or (3) the server or device supplied a default value. An
398 agent SHALL represent an attribute as an entry (row) in the Attribute table in this MIB in
399 which entries are present only when necessary. Attributes are identified in this MIB by an
400 enum.
- 401 Job Monitoring (using SNMP): the activity of a management application of accessing the
402 MIB and (1) identifying jobs in the job tables being processed by the server, printer or
403 other devices, and (2) displaying information to the user about the processing of the job.
- 404 Job Accounting: the activity of a management application of accessing the MIB and
405 recording what happens to the job during and after the processing of the job.

406 **2.1 System Configurations for the Job Monitoring MIB**

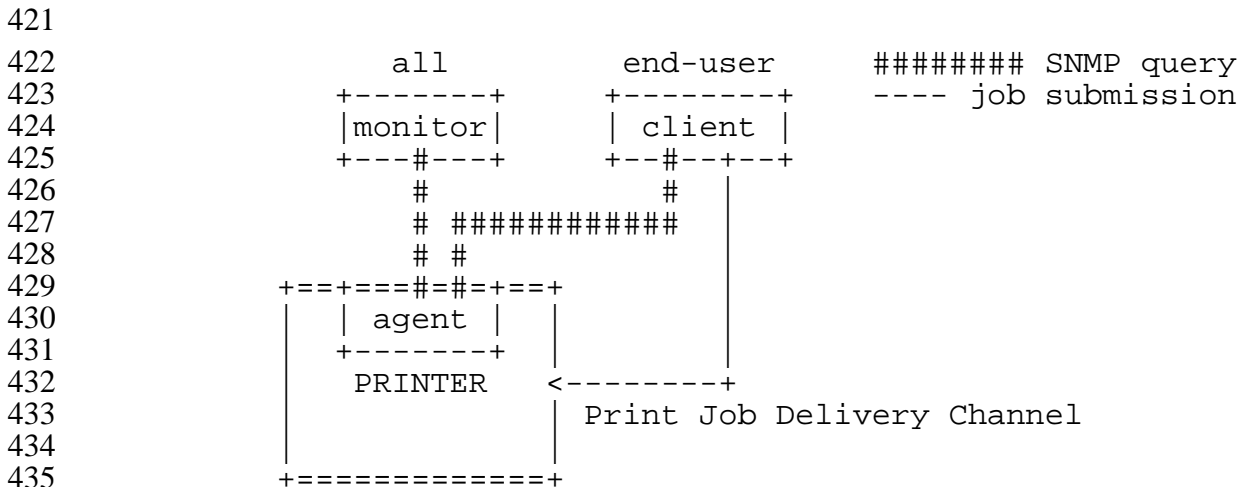
407 This section enumerates the three configurations in which the Job Monitoring MIB is
 408 intended to be used. To simplify the pictures, the *devices* are shown as *printers*. See
 409 Goals section.

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

413 **2.1.1 Configuration 1 - client-printer**

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

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



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

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

- 439 1. Multiple **clients** MAY submit jobs to a **printer**.
- 440 2. Multiple **clients** MAY monitor a **printer**.
- 441 3. Multiple **monitors** MAY monitor a **printer**.
- 442 4. A **client** MAY submit jobs to multiple **printers**.
- 443 5. A **monitor** MAY monitor multiple **printers**.

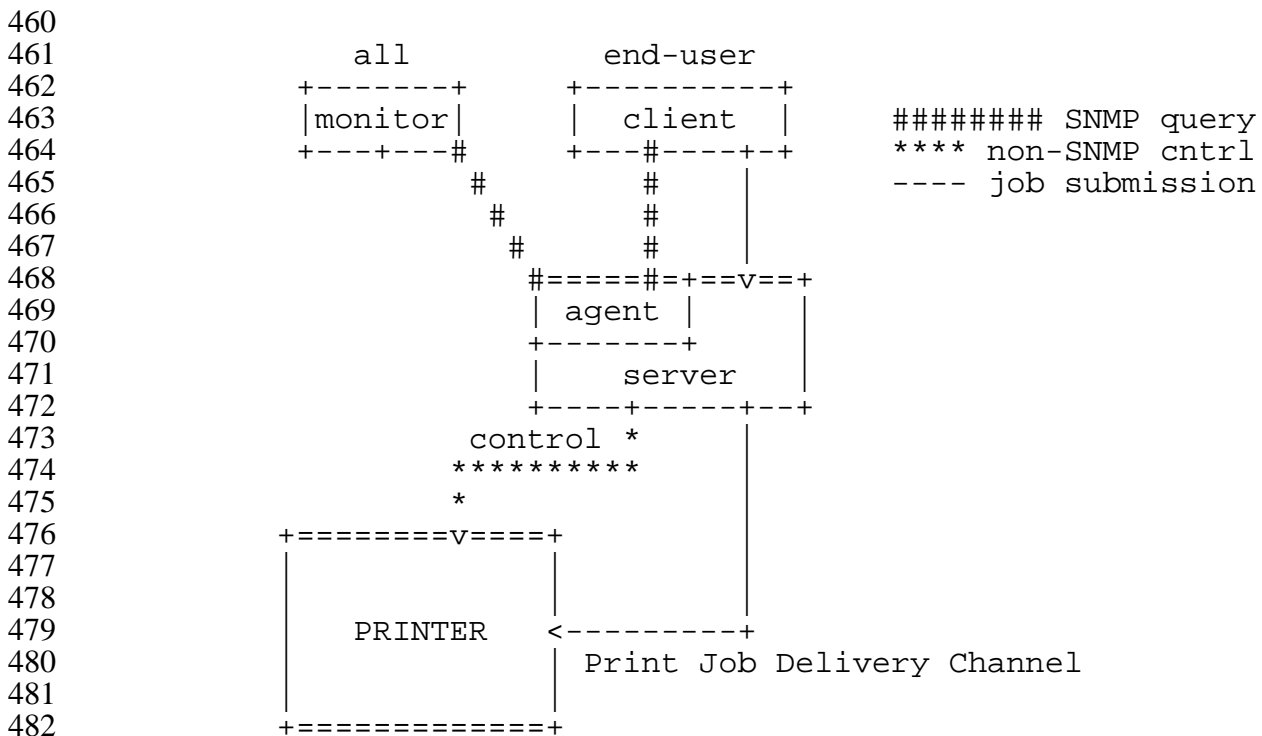
444 **2.1.2 Configuration 2 - client-server-printer - agent in the server**

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

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

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

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



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

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

- 486 1. Multiple **clients** MAY submit jobs to a **server**.
487 2. Multiple **clients** MAY monitor a **server**.
488 3. Multiple **monitors** MAY monitor a **server**.
489 4. A **client** MAY submit jobs to multiple **servers**.
490 5. A **monitor** MAY monitor multiple **servers**.
491 6. Multiple **servers** MAY submit jobs to a **printer**.
492 7. Multiple **servers** MAY control a **printer**.

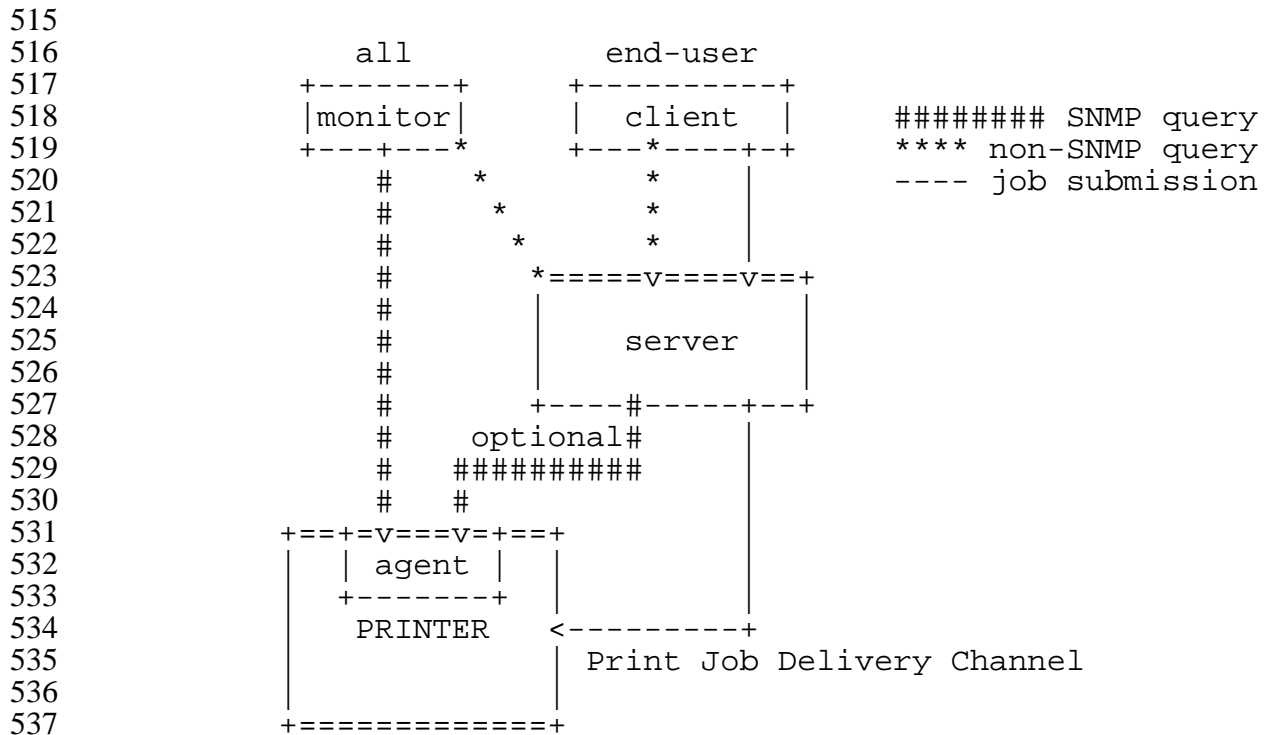
493 **2.1.3 Configuration 3 - client-server-printer - client monitors printer agent and**
494 **server**

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

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

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

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



538 **Figure 2-3 - Configuration 3 - client-server-printer - client monitors printer agent**
539 **and server**

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

- 542 1. Multiple **clients** MAY submit jobs to a **server**.
- 543 2. Multiple **clients** MAY monitor a **server**.
- 544 3. Multiple **monitors** MAY monitor a **server**.
- 545 4. A **client** MAY submit jobs to multiple **servers**.
- 546 5. A **monitor** MAY monitor multiple **servers**.
- 547 6. Multiple **servers** MAY submit jobs to a **printer**.
- 548 7. Multiple **servers** MAY control a **printer**.

549 **3. Managed Object Usage**

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

551 **3.1 Conformance Considerations**

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

555 **3.1.1 Conformance Terminology**

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

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

569 **3.1.2 Agent Conformance Requirements**

570 A conforming agent:

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

579 NOTE - This MIB, like the Printer MIB, is written following the subset of SMIV2 that
580 can be supported by SMIV1 and SNMPv1 implementations.

581 **3.1.2.1 MIB II System Group objects**

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

584 **3.1.2.2 MIB II Interface Group objects**

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

587 3.1.2.3 Printer MIB objects

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

594 **3.1.3 Job Monitoring Application Conformance Requirements**

595 A conforming job monitoring application:

- 596 1. SHALL accept the full syntactic range for all objects in all MANDATORY groups and
597 all MANDATORY attributes that are required to be implemented by an agent
598 according to Section 3.1.2 and SHALL either present them to the user or ignore them.
- 599 2. SHALL accept the full syntactic range for *all* attributes, including enum and bit values
600 specified in this specification and additional ones that may be registered with IANA
601 and SHALL either present them to the user or ignore them. In particular, a
602 conforming job monitoring application SHALL not malfunction when receiving any
603 standard or registered enum or bit values. See Section 3.6 entitled "IANA
604 Considerations".
- 605 3. SHALL NOT fail when operating with agents that materialize attributes *after* the job
606 has been submitted, as opposed to when the job is submitted.
- 607 4. SHALL, if it supports a time attribute, accept either form of the time attribute, since
608 agents are free to implement either time form.

609 **3.2 The Job Tables and the Oldest Active and Newest Active Indexes**

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

- 612 1. **jmGeneralJobSetIndex** - which job set
- 613 2. **jmJobIndex** - which job in the job set

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

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

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

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

630 Each time a new job is accepted by the server or device that the agent is providing access
631 to AND that job is to be 'active' (**pending**, **processing**, or **processingStopped**, but not
632 **pendingHeld**), the agent SHALL copy the value of the job's **jmJobIndex** to the
633 **jmGeneralNewestActiveJobIndex** object. If the new job is to be 'inactive'
634 (**pendingHeld** state), the agent SHALL not change the value of
635 **jmGeneralNewestActiveJobIndex** object.

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

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

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

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

655 If an application detects that the **jmGeneralNewestActiveJobIndex** is smaller than
656 **jmGeneralOldestActiveJobIndex**, the job index has wrapped. In this case, the

657 application SHALL reset the index to **1** when the end of the table is reached and continue
658 the GetNext operations to find the rest of the active jobs.

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

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

665 3.3 The Attribute Mechanism

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

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

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

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

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

691 3.3.1 Conformance of Attribute Implementation

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

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

700 3.3.2 Useful, 'Unknown', and 'Other' Values for Objects and Attributes

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

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

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

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

726 **3.3.3 Data Sub-types and Attribute Naming Conventions**

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

739 NOTE: The Table of Contents also lists the data sub-type and/or data sub-types of each
 740 attribute, using the textual-convention name when such is defined. The following
 741 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.
'Octets63'	OCTET STRING(SIZE(0..63))
'Octets(m..n)'	For all other OCTET STRING ranges, the exact range is indicated.

742 **3.3.4 Single-Value (Row) Versus Multi-Value (MULTI-ROW) Attributes**

743 Most attributes SHALL have only one row per job. However, a few attributes can have
 744 multiple values per job or even per document, where each value is a separate row in the
 745 **jmAttributeTable**. Unless indicated with '**MULTI-ROW:**' in the **JmAttributeTypeTC**
 746 description, an agent SHALL ensure that each attribute occurs only once in the
 747 **jmAttributeTable** for a job. Most of the '**MULTI-ROW**' attributes do not allow
 748 duplicate values, i.e., the agent SHALL ensure that each value occurs only once for a job.
 749 Only if the specification of the '**MULTI-ROW**' attribute also says "the values NEED NOT
 750 be unique" can the agent allow duplicate values to occur for the job.

751 NOTE - Duplicate are allowed for 'extensive' '**MULTI-ROW**' attributes, such as
 752 **fileName(34)** or **documentName(35)**, but are not allowed for 'intensive' '**MULTI-ROW**'
 753 attributes, such as **mediumConsumed(171)** and **documentFormat(38)**.

754 **3.3.5 Requested Attributes**

755 A number of attributes record requirements for the job. Such attribute names end with the
756 word '**Requested**'. In the interests of brevity, the phrase 'requested' SHALL mean: (1)
757 requested by the client (or intervening server) in the job submission protocol and MAY
758 also mean (2) embedded in the submitted document data, and/or (3) defaulted by the
759 recipient device or server with the same semantics as if the requester had supplied,
760 depending on implementation.

761 **3.3.6 Consumption Attributes**

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

768 **3.3.7 Index Value Attributes**

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

775 **3.4 Job Identification**

776 There are a number of attributes that permit a user, operator or system administrator to
777 identify jobs of interest, such as **jobName**, **jobOriginatingHost**, etc. In addition, there is
778 a Job Submission ID object that allows a monitoring application to quickly locate and
779 identify a particular job of interest that was submitted from a particular client by the user
780 invoking the monitoring application. The Job Monitoring MIB needs to provide for
781 identification of the job at both sides of the job submission process. The primary
782 identification point is the client side. The Job Submission ID allows the monitoring
783 application to identify the job of interest from all the jobs currently "known" by the server
784 or device. The Job Submission ID can be assigned by either the client's local system or a
785 downstream server or device. The point of assignment depends on the job submission
786 protocol in use.

787 The server/device-side identifier, called the **jmJobIndex** object, SHALL be assigned by
788 the SNMP Job Monitoring MIB agent when the server or device accepts the jobs from

789 submitting clients. The **jmJobIndex** object allows the interested party to obtain all
790 objects desired that relate to this job. The MIB provides a mapping table that maps each
791 Job Submission ID (generated by the client) to the corresponding **jmJobIndex** value
792 generated by the agent, so that an application can determine the correct value for the
793 **jmJobIndex** value for the job of interest in a single Get operation, given the Job
794 Submission ID. See the **jmJobIDGroup**.

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

798 **3.5 Internationalization Considerations**

799 There are a number of objects in this MIB that are represented as coded character sets
800 with a data type of **OCTET STRING**. Most of the objects are supplied as job attributes
801 by the client that submits the job to the server or device and so are represented in the
802 coded character set specified by that client.

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

811 This MIB also contains objects that are represented using the **DateAndTime** textual
812 convention from SMIV2 [SMIV2-TC]. The job management application SHALL display
813 such objects in the locale of the user running the monitoring application.

814 **3.6 IANA Considerations**

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

821 **3.6.1 IANA Registration of enums**

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

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

833 3.6.1.1 Type 1 enumerations

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

836 There are no type 1 enums in the current draft.

837 3.6.1.2 Type 2 enumerations

838 Type 2 enumeration: An initial set of values are defined in the Job Monitoring MIB
839 specification. Additional enumerated values are registered after review by this working
840 group or an editor appointed by IANA after this working group is no longer active.

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

- 842 **1. JmTimeStampTC**
- 843 **2. JmFinishingTC** [same enum values as IPP "finishing" attribute]
- 844 **3. JmPrintQualityTC** [same enum values as IPP "print-quality" attribute]
- 845 **4. JmTonerEconomyTC**
- 846 **5. JmMediumTypeTC**
- 847 **6. JmJobSubmissionTypeTC**
- 848 **7. JmJobStateTC** [same enum values as IPP "job-state" attribute]
- 849 **8. JmAttributeTypeTC**

850 For those textual conventions that have the same enum values as the indicated IPP Job
851 attribute SHALL be simultaneously registered by IANA for use with IPP [ipp-model] and
852 the Job Monitoring MIB.

853 3.6.1.3 Type 3 enumeration

854 Type 3 enumeration: An initial set of values are defined in the Job Monitoring MIB
855 specification. Additional enumerated values are registered through IANA without
856 working group review.

857 There are no type 3 enums in the current draft.

858 3.6.2 IANA Registration of type 2 bit values

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

- 860 1. **JmJobServiceTypesTC**
- 861 2. **JmJobStateReasons1TC**
- 862 3. **JmJobStateReasons2TC**
- 863 4. **JmJobStateReasons3TC**
- 864 5. **JmJobStateReasons4TC**

865 These textual-conventions are defined as bits in an Integer so that they can be used with
866 SNMPv1 SMI. The **jobStateReasonsN** ($N=1..4$) attributes are defined as bit values using
867 the corresponding **JmJobStateReasonsNTC** textual-conventions.

868 The registration of **JmJobServiceTypesTC** and **JmJobStateReasonsNTC** bit values
869 SHALL follow the procedures for a type 2 enum as specified in Section 3.6.1.2.

870 3.6.3 IANA Registration of Job Submission Id Formats

871 In addition to enums and bit values, this specification assigns a single ASCII digit or letter
872 to various job submission ID formats. See the **JmJobSubmissionIDTypeTC** textual-
873 convention and the object. The registration of **jmJobSubmissionID** format numbers
874 SHALL follow the procedures for a type 2 enum as specified in Section 3.6.1.2.

875 3.6.4 IANA Registration of MIME types/sub-types for document-formats

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

880 3.7 Security Considerations

881 3.7.1 Read-Write objects

882 All objects are read-only, greatly simplifying the security considerations. If another MIB
883 augments this MIB, that MIB might accept SNMP Write operations to objects in that

884 MIB whose effect is to modify the values of read-only objects in this MIB. However, that
885 MIB SHALL have to support the required access control in order to achieve security, not
886 this MIB.

887 **3.7.2 Read-Only Objects In Other User's Jobs**

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

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

899 **3.8 Notifications**

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

904 **4. MIB specification**

905 The following pages constitute the actual Job Monitoring MIB.

```

906 Job-Monitoring-MIB DEFINITIONS ::= BEGIN
907
908 IMPORTS
    MODULE-IDENTITY, OBJECT-TYPE, experimental, Integer32
    TEXTUAL-CONVENTION
    MODULE-COMPLIANCE, OBJECT-GROUP
    FROM SNMPv2-SMI
    FROM SNMPv2-TC
    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
    FROM Printer-MIB

909 -- Use the experimental (54) OID assigned to the Printer MIB[print-mib]
910 -- before it was published as RFC 1759.
911 -- Upon publication of the Job Monitoring MIB as an RFC, delete this
912 -- comment and the line following this comment and change the
913 -- reference of { temp 105 } (below) to { mib-2 X }.
914 -- This will result in changing:
915 -- 1 3 6 1 3 54 jobmonMIB(105)  to:
916 -- 1 3 6 1 2 1 jobmonMIB(X)
917 -- This will make it easier to translate prototypes to
918 -- the standard namespace because the lengths of the OIDs won't
919 -- change.
920 temp OBJECT IDENTIFIER ::= { experimental 54 }
921
922 jobmonMIB MODULE-IDENTITY
923     LAST-UPDATED "9707210000Z"
924     ORGANIZATION "IETF Printer MIB Working Group"
925     CONTACT-INFO
926         "Tom Hastings
927         Postal: Xerox Corp.
928         Mail stop ESAE-231
929         701 S. Aviation Blvd.
930         El Segundo, CA 90245
931
932         Tel: (301)333-6413
933         Fax: (301)333-5514
934         E-mail: hstings@cp10.es.xerox.com
935
936         Send comments to the printmib WG using the Job Monitoring
937         Project (JMP) Mailing List: jmp@pwg.org
938
939         To learn how to subscribe to the JMP mailing list,
940         send email to: jmp-request@pwg.org
941
942         For further information, access the PWG web page under 'JMP':
943

```

```

944         http://www.pwg.org/"
945     DESCRIPTION
946         "The MIB module for monitoring job in servers, printers, and other devices.
947
948         File: draft-ietf-printmib-job-monitor-04.txt
949         Version: 0.84"
950     ::= { temp 105 }
951
952
953
954 -- Textual conventions for this MIB module
955
956
957 JmTimeStampTC ::= TEXTUAL-CONVENTION
958     STATUS      current
959     DESCRIPTION
960         "The simple time at which an event took place.  The units SHALL be in seconds since the
961         system was booted.
962
963         NOTE - JmTimeStampTC is defined in units of seconds, rather than 100ths of seconds, so as
964         to be simpler for agents to implement (even if they have to implement the 100ths of a second to
965         comply with implementing sysUpTime in MIB-II[mib-II].)
966
967         NOTE - JmTimeStampTC is defined as an Integer32 so that it can be used as a value of an
968         attribute, i.e., as a value of the jmAttributeValueAsInteger object.  The TimeStamp textual-
969         convention defined in SMN Pv2-TC is defined as an APPLICATION 3 IMPLICIT INTEGER
970         tag, not an Integer32, so cannot be used in this MIB as one of the values of
971         jmAttributeValueAsInteger."
972     SYNTAX      INTEGER(0..2147483647)
973
974
975
976
977 JmJobSourcePlatformTypeTC ::= TEXTUAL-CONVENTION
978     STATUS      current
979     DESCRIPTION
980         "The source platform type that can submit jobs to servers or devices in any of the 3
981         configurations."
982     REFERENCE
983         "This is a type 2 enumeration.  See Section 3.6.1.2."
984     SYNTAX      INTEGER {
985         other(1),
986         unknown(2),
987         sptUNIX(3),           -- UNIX(tm)
988         sptOS2(4),         -- OS/2
989         sptPCDOS(5),      -- DOS
990         sptNT(6),         -- NT

```

```

985     }
986
987
988
989
990
991 JmFinishingTC ::= TEXTUAL-CONVENTION
992     STATUS      current
993     DESCRIPTION
994         "The type of finishing operation.
995
996         These values are the same as the enum values of the IPP 'finishings' attribute. See Section
997         3.6.1.2.
998
999         other(1),
1000             Some other finishing operation besides one of the specified or registered values.
1001
1002         unknown(2),
1003             The finishing is unknown.
1004
1005         none(3),
1006             Perform no finishing.
1007
1008         staple(4),
1009             Bind the document(s) with one or more staples. The exact number and placement of the
1010             staples is site-defined.
1011
1012         stapleTopLeft(5),
1013             Place one or more staples on the top left corner of the document(s).
1014
1015         stapleBottomLeft(6),
1016             Place one or more staples on the bottom left corner of the document(s).
1017
1018         stapleTopRight(7),
1019             Place one or more staples on the top right corner of the document(s).
1020
1021         stapleBottomRight(8),
1022             Place one or more staples on the bottom right corner of the document(s).
1023
1024         saddleStitch(9),
1025             Bind the document(s) with one or more staples (wire stitches) along the middle fold. The
1026             exact number and placement of the stitches is site-defined.

```

1027
 1028 **edgeStitch(10),**
 1029 Bind the document(s) with one or more staples (wire stitches) along one edge. The exact
 1030 number and placement of the staples is site-defined.
 1031
 1032 **punch(11),**
 1033 This value indicates that holes are required in the finished document. The exact number
 1034 and placement of the holes is site-defined. The punch specification MAY be satisfied (in a
 1035 site- and implementation-specific manner) either by drilling/punching, or by substituting
 1036 pre-drilled media.
 1037
 1038 **cover(12),**
 1039 This value is specified when it is desired to select a non-printed (or pre-printed) cover for
 1040 the document. This does not supplant the specification of a printed cover (on cover stock
 1041 medium) by the document itself.
 1042
 1043 **bind(13)**
 1044 This value indicates that a binding is to be applied to the document; the type and
 1045 placement of the binding is product-specific."
 1046 REFERENCE
 1047 "This is a type 2 enumeration. See Section 3.6.1.2."
 1048 SYNTAX INTEGER {
 1049 other(1),
 1050 unknown(2),
 1051 none(3),
 1052 staple(4),
 1053 stapleTopLeft(5),
 1054 stapleBottomLeft(6),
 1055 stapleTopRight(7),
 1056 stapleBottomRight(8),
 1057 saddleStitch(9),
 1058 edgeStitch(10),
 1059 punch(11),
 1060 cover(12),
 1061 bind(13)
 1062 }
 1063
 1064
 1065
 1066
 1067
 1068 **JmPrintQualityTC ::= TEXTUAL-CONVENTION**
 1069 STATUS current
 1070 DESCRIPTION
 1071 "Print quality settings."
 1072
 1073 These values are the same as the enum values of the IPP 'print-quality' attribute. See Section
 1074 3.6.1.2."


```

1075 REFERENCE
1076     "This is a type 2 enumeration. See Section 3.6.1.2."
1077 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.
    }
1078
1079
1080
1081
1082
1083 JmPrinterResolutionTC ::= TEXTUAL-CONVENTION
1084     STATUS      current
1085     DESCRIPTION
1086         "Printer resolutions.
1087
1088         Nine octets consisting of two 4-octet SIGNED-INTEGERS followed by a SIGNED-BYTE.
1089         The values are the same as those specified in the Printer MIB [printmib]. The first SIGNED-
1090         INTEGER contains the value of prtMarkerAddressabilityXFeedDir. The second SIGNED-
1091         INTEGER contains the value of prtMarkerAddressabilityFeedDir. The SIGNED-BYTE
1092         contains the value of prtMarkerAddressabilityUnit.
1093
1094         Note: the latter value is either 3 (tenThousandsOfInches) or 4 (micrometers) and the
1095         addressability is in 10,000 units of measure. Thus the SIGNED-INTEGERS represent integral
1096         values in either dots-per-inch or dots-per-centimeter.
1097
1098         The syntax is the same as the IPP 'printer-resolution' attribute. See Section 3.6.1.2."
1099 SYNTAX  OCTET STRING (SIZE(9))
1100
1101
1102
1103
1104
1105 JmTonerEconomyTC ::= TEXTUAL-CONVENTION
1106     STATUS      current
1107     DESCRIPTION
1108         "Toner economy settings."
1109     REFERENCE
1110         "This is a type 2 enumeration. See Section 3.6.1.2."
1111 SYNTAX  INTEGER {
        unknown(2),   -- unknown.
        off(3),       -- Off. Normal. Use full toner.
        on(4)         -- On. Use less toner than normal.
    }

```

```

1112     }
1113
1114
1115
1116
1117
1118 JmBooleanTC ::= TEXTUAL-CONVENTION
1119     STATUS     current
1120     DESCRIPTION
1121         "Boolean true or false value."
1122     REFERENCE
1123         "This is a type 2 enumeration.  See Section 3.6.1.2."
1124     SYNTAX     INTEGER {
1125
1126         unknown(2),           -- unknown.
1127         false(3),             -- FALSE.
1128         true(4)              -- TRUE.
1129     }
1130
1131 JmMediumTypeTC ::= TEXTUAL-CONVENTION
1132     STATUS     current
1133     DESCRIPTION
1134         "Identifies the type of medium.
1135
1136         other(1),
1137             The type is neither one of the values listed in this specification nor a registered value.
1138
1139         unknown(2),
1140             The type is not known.
1141
1142         stationery(3),
1143             Separately cut sheets of an opaque material.
1144
1145         transparency(4),
1146             Separately cut sheets of a transparent material.
1147
1148         envelope(5),
1149             Envelopes that can be used for conventional mailing purposes.
1150
1151         envelopePlain(6),
1152             Envelopes that are not preprinted and have no windows.
1153
1154         envelopeWindow(7),
1155             Envelopes that have windows for addressing purposes.

```

1156
 1157 **continuousLong(8),**
 1158 Continuously connected sheets of an opaque material connected along the long edge.
 1159
 1160 **continuousShort(9),**
 1161 Continuously connected sheets of an opaque material connected along the short edge.
 1162
 1163 **tabStock(10),**
 1164 Media with tabs.
 1165
 1166 **multiPartForm(11),**
 1167 Form medium composed of multiple layers not pre-attached to one another; each sheet
 1168 MAY be drawn separately from an input source.
 1169
 1170 **labels(12),**
 1171 Label-stock.
 1172
 1173 **multiLayer(13)**
 1174 Form medium composed of multiple layers which are pre-attached to one another, e.g. for
 1175 use with impact printers."
 1176 REFERENCE
 1177 "This is a type 2 enumeration. See Section 3.6.1.2."
 1178 SYNTAX INTEGER {
 1179 other(1),
 1180 unknown(2),
 1181 stationery(3),
 1182 transparency(4),
 1183 envelope(5),
 1184 envelopePlain(6),
 1185 envelopeWindow(7),
 1186 continuousLong(8),
 1187 continuousShort(9),
 1188 tabStock(10),
 1189 multiPartForm(11),
 1190 labels(12),
 1191 multiLayer(13)
 1192 }
 1193
 1194
 1195
 1196
 1197
 1198 **JmJobSubmissionTypeTC ::= TEXTUAL-CONVENTION**
 1199 STATUS current
 1200 DESCRIPTION
 1201 "Identifies the format type of a job submission ID."
 1202
 1203 The ASCII characters '0-9', 'A-Z', and 'a-z' are assigned in order giving 62 possible formats.

1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252

Each job submission ID is a fixed-length, 48-octet printable ASCII coded character string, consisting of the following fields:

- octet 1 The format letter.
- octets 2-40 A 39-character, 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 number to make the ID quasi-unique.

If the client does not supply a job submission ID in the job submission protocol, then the server SHALL assign a job submission ID using any of the standard formats that are reserved to the agent. Clients SHALL not use formats that are reserved to agents.

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

Format Letter	Description
-----	-----
'0'	octets 2-40: last 39 bytes of the jmJobOwner object. octets 41-48: 8-decimal-digit sequential number This format is reserved to agents for use when the client does not supply a job submission ID. Clients wishing to use a job submission ID that incorporates the job owner, SHALL use format '8'.
NOTE - other formats may be registered that are reserved to the agent for use when the client does not supply a job submission ID.	
'1'	octets 2-40: last 39 bytes of the jobName attribute. octets 41-48: 8-decimal-digit random number
'2'	octets 2-40: 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: 8-decimal-digit sequential number
'3'	octets 2-40: last 39 bytes of the client URL [URI-spec]. octets 41-48: 8-decimal-digit sequential number
'4'	octets 2-40: 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: 8-decimal-digit sequential number

- 1253
- 1254 '5' octets 2-40: last 39 bytes of a user number, such
- 1255 as POSIX user number.
- 1256 octets 41-48: 8-decimal-digit sequential number
- 1257
- 1258 '6' octets 2-40: last 39 bytes of the user account
- 1259 number.
- 1260 octets 41-48: 8-decimal-digit sequential number
- 1261
- 1262 '7' octets 2-40: last 39 bytes of the DTMF incoming
- 1263 FAX routing number.
- 1264 octets 41-48: 8-decimal-digit sequential number
- 1265
- 1266 '8' octets 2-40: last 39 bytes of the job owner name
- 1267 (that the agent returns in the **jmJobOwner** object).
- 1268 octets 41-48: 8-decimal-digit sequential number
- 1269

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."

1281 REFERENCE

1282 "This is like a type 2 enumeration. See section 3.6.3."

1283 SYNTAX OCTET STRING(SIZE(1)) -- ASCII '0'-'9', 'A'-'Z', 'a'-'z'

1284
1285
1286
1287
1288

1289 **JmJobStateTC** ::= TEXTUAL-CONVENTION

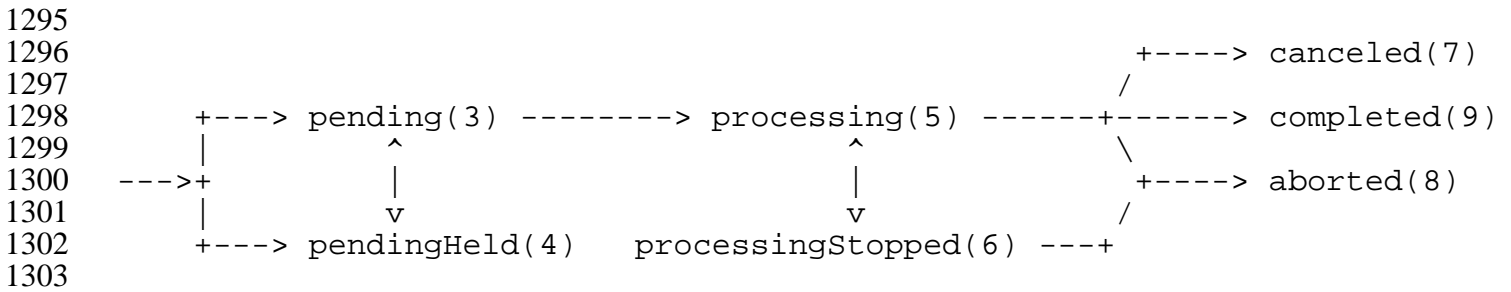
1290 STATUS current

1291 DESCRIPTION

1292 "The current state of the job (**pending, processing, completed**, etc.).

1293
1294

The following figure shows the normal job state transitions:



1304 **Figure 4 - Normal Job State Transitions**

1305 Normally a job progresses from left to right. Other state transitions are unlikely, but are not
1306 forbidden. Not shown are the transitions to the **canceled** state from the **pending**,
1307 **pendingHeld**, **processing**, and **processingStopped** states.

1308 Jobs in the **pending**, **processing**, and **processingStopped** states are called 'active', while jobs in
1309 the **pendingHeld**, **canceled**, **aborted**, and **completed** are called 'inactive'.

1310 These values are the same as the enum values of the IPP 'job-state' job attribute. See Section
1311 3.6.1.2.

1312 **other(1),**
1313 The job state is *not* one of the defined states.

1314 **unknown(2),**
1315 The job state is *not* known, or its state is indeterminate.

1316 **pending(3),**
1317 The job is a candidate to start processing, but is not yet processing.

1318 **pendingHeld(4),**
1319 The job is not a candidate for processing for any number of reasons but will return to the
1320 pending state as soon as the reasons are no longer present. The job's
1321 **jmJobStateReasons1** object and/or **jobStateReasonsN** ($N=2..4$) attributes SHALL
1322 indicate why the job is no longer a candidate for processing. The reasons are represented
1323 as bits in the **jmJobStateReasons1** object and/or **jobStateReasonsN** ($N=2..4$) attributes.
1324 See the **JmJobStateReasonsNTC** ($N=1..4$) textual convention for the specification of
1325 each reason.

1326 **processing(5),**
1327 Either:
1328 1. The job is using, or is attempting to use, one or more document transforms which
1329 include (1) purely software processes that are interpreting a PDL, and (2) hardware
1330 devices that are interpreting a PDL, making marks on a medium, and/or performing
1331 finishing, such as stapling, etc.

1332 OR

1343
 1344 2. (configuration 2) the server has made the job ready for printing, but the output device is
 1345 not yet printing it, either because the job hasn't reached the output device or because the
 1346 job is queued in the output device or some other spooler, awaiting the output device to
 1347 print it.
 1348
 1349 When the job is in the **processing** state, the entire job state includes the detailed status
 1350 represented in the device MIB indicated by the **hrDeviceIndex** value of the job's
 1351 **physicalDevice** attribute, if the agent implements such a device MIB.
 1352
 1353 Implementations MAY, though they NEED NOT, include additional values in the job's
 1354 **jmJobStateReasons1** object to indicate the progress of the job, such as adding the
 1355 **jobPrinting** value to indicate when the device is actually making marks on a medium.
 1356
 1357 **processingStopped(6),**
 1358 The job has stopped while processing for any number of reasons and will return to the
 1359 **processing** state as soon as the reasons are no longer present.
 1360
 1361 The job's **jmJobStateReasons1** object and/or the job's **jobStateReasonsN** ($N=2..4$)
 1362 attributes MAY indicate why the job has stopped processing. For example, if the output
 1363 device is stopped, the **deviceStopped** value MAY be included in the job's
 1364 **jmJobStateReasons1** object.
 1365
 1366 NOTE - When an output device is stopped, the device usually indicates its condition in
 1367 human readable form at the device. The management application can obtain more
 1368 complete device status remotely by querying the appropriate device MIB using the job's
 1369 **deviceIndex** attribute(s), if the agent implements such a device MIB
 1370
 1371 **canceled(7),**
 1372 A client has canceled the job and the job is either: (1) in the process of being terminated by
 1373 the server or device or (2) has completed terminating. The job's **jmJobStateReasons1**
 1374 object SHOULD contain either the **canceledByUser** or **canceledByOperator** value.
 1375
 1376 **aborted(8),**
 1377 The job has been aborted by the system, usually while the job was in the processing or
 1378 processingStopped state.
 1379
 1380 **completed(9)**
 1381 The job has completed successfully or with warnings or errors after processing and all of
 1382 the media have been successfully stacked in the appropriate output bin(s). The job's
 1383 **jmJobStateReasons1** object SHOULD contain one of: **completedSuccessfully**,
 1384 **completedWithWarnings**, or **completedWithErrors** values."
 1385 REFERENCE
 1386 "This is a type 2 enumeration. See Section 3.6.1.2."
 1387 SYNTAX INTEGER {
 1388 other(1),
 1389 unknown(2),
 1390 pending(3),
 1391 pendingHeld(4),

1392 processing(5),
 1393 processingStopped(6),
 1394 canceled(7),
 1395 aborted(8),
 1396 completed(9)
 1397 }
 1398

1399
 1400 **JmAttributeTypeTC ::= TEXTUAL-CONVENTION**

1401 STATUS current

1402 DESCRIPTION

1403 "The type of the attribute which identifies the attribute.
 1404

1405 In the following definitions of the enums, each description indicates whether the useful value of
 1406 the attribute SHALL be represented using the **jmAttributeValueAsInteger** or the
 1407 **jmAttributeValueAsOctets** objects by the initial tag: 'INTEGER:' or 'OCTETS:',
 1408 respectively.
 1409

1410 Some attributes allow the agent implementer a choice of useful values of either an integer, an
 1411 octets representation, or both, depending on implementation. These attributes are indicated with
 1412 'INTEGER:' AND/OR 'OCTETS:' tags.
 1413

1414 A very few attributes require both objects at the same time to represent a pair of useful values
 1415 (see **mediumConsumed(171)**). These attributes are indicated with 'INTEGER:' AND
 1416 'OCTETS:' tags. See the **jmAttributeGroup** for the descriptions of these two MANDATORY
 1417 objects.
 1418

1419 NOTE - The enum assignments are grouped logically with values assigned in groups of 20, so
 1420 that additional values may be registered in the future and assigned a value that is part of their
 1421 logical grouping.
 1422

1423 NOTE: No attribute name exceeds 31 characters.
 1424

1425 The standard attribute types defined at the time of completion of the specification are:

1426 **jmAttributeTypeIndex**

Datatype

1427 -----

1428
 1429 **other(1),**

Integer32(-2..2147483647)

AND/OR

OCTET STRING(SIZE(0..63))

1430 INTEGER: and/or OCTETS: An attribute that is not in the list and/or that has not been
 1431 approved and registered with IANA.
 1432

1433 ++++++

1434 + **Job State attributes**

1435 +
 1436
 1437
 1438
 1439

1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486

+ **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), **OCTET STRING(SIZE(0..63))**
 OCTETS: MULTI-ROW: A coded character set message that is generated during the processing of the job as a simple form of processing log to show progress and any problems.

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

+++++

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

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

 NOTE: This attribute NEED NOT be printable characters.

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

1487 NOTE - This attribute is intended for enabling a user to find his/her job that a server
 1488 submitted to a device when either the client does not support the **jmJobSubmissionID** or
 1489 the server does not pass the **jmJobSubmissionID** through to the device.
 1490

1491 **jobName(23),** **OCTET STRING(SIZE(0..63))**

1492 OCTETS: The human readable string name of the job as assigned by the submitting user
 1493 to help the user distinguish between his/her various jobs. This name does not need to be
 1494 unique.
 1495

1496 This attribute is intended for enabling a user or the user's application to convey a job name
 1497 that MAY be printed on a start sheet, returned in a **query** result, or used in notification or
 1498 logging messages.
 1499

1500 In order to assist users to find their jobs for job submission protocols that don't supply a
 1501 **jmJobSubmissionID**, the agent SHOULD maintain the **jobName** attribute for the time
 1502 specified by the **jmGeneralJobPersistence** object, rather than the (shorter)
 1503 **jmGeneralAttributePersistence** object.
 1504

1505 If this attribute is not specified when the job is submitted, no job name is assumed, but
 1506 implementation specific defaults are allowed, such as the value of the **documentName**
 1507 attribute of the first document in the job or the **fileName** attribute of the first document in
 1508 the job.
 1509

1510 The **jobName** attribute is distinguished from the **jobComment** attribute, in that the
 1511 **jobName** attribute is intended to permit the submitting user to distinguish between
 1512 different jobs that he/she has submitted. The **jobComment** attribute is intended to be free
 1513 form additional information that a user might wish to use to communicate with
 1514 himself/herself, such as a reminder of what to do with the results or to indicate a different
 1515 set of input parameters were tried in several different job submissions.
 1516

1517 **jobServiceTypes(24),** **JmJobServiceTypesTC**

1518 INTEGER: Specifies the type(s) of service to which the job has been submitted (print,
 1519 fax, scan, etc.). The service type is bit encoded with each job service type so that more
 1520 general and arbitrary services can be created, such as services with more than one
 1521 destination type, or ones with only a source or only a destination. For example, a job
 1522 service might **scan**, **faxOut**, and **print** a single job. In this case, three bits would be set in
 1523 the **jobServiceTypes** attribute, corresponding to the hexadecimal values: **0x8** + **0x20** +
 1524 **0x4**, respectively, yielding: **0x2C**.
 1525

1526 Whether this attribute is set from a job attribute supplied by the job submission client or is
 1527 set by the recipient job submission server or device depends on the job submission
 1528 protocol. This attribute SHALL be implemented if the server or device has other types in
 1529 addition to or instead of printing.
 1530

1531 One of the purposes of this attribute is to permit a requester to filter out jobs that are not
 1532 of interest. For example, a printer operator may only be interested in jobs that include
 1533 printing.
 1534

1535 **jobSourceChannelIndex(25),** Integer32(0..2147483647)
 1536 INTEGER: The index of the row in the associated Printer MIB[print-mib] of the channel
 1537 which is the source of the print job.
 1538

1539 **jobSourcePlatformType(26),** **JmJobSourcePlatformTypeTC**
 1540 INTEGER: The source platform type of the immediate upstream submitter that submitted
 1541 the job to the server (configuration 2) or device (configuration 1 and 3) to which the agent
 1542 is providing access. For configuration 1, this is the type of the client that submitted the
 1543 job to the device; for configuration 2, this is the type of the client that submitted the job
 1544 to the server; and for configuration 3, this is the type of the server that submitted the job
 1545 to the device.
 1546

1547 **submittingServerName(27),** **OCTET STRING(SIZE(0..63))**
 1548 OCTETS: For configuration 3 only: The administrative name of the server that submitted
 1549 the job to the device.
 1550

1551 **submittingApplicationName(28),** **OCTET STRING(SIZE(0..63))**
 1552 OCTETS: The name of the client application (not the server in configuration 3) that
 1553 submitted the job to the server or device.
 1554

1555 **jobOriginatingHost(29),** **OCTET STRING(SIZE(0..63))**
 1556 OCTETS: The name of the client host (not the server host name in configuration 3) that
 1557 submitted the job to the server or device.
 1558

1559 **deviceNameRequested(30),** **OCTET STRING(SIZE(0..63))**
 1560 OCTETS: The administratively defined coded character set name of the target device
 1561 requested by the submitting user. For configuration 1, its value corresponds to the Printer
 1562 MIB[print-mib]: **prtGeneralPrinterName** object. For configuration 2 and 3, its value is
 1563 the name of the logical or physical device that the user supplied to indicate to the server
 1564 on which device(s) they wanted the job to be processed.
 1565

1566 **queueNameRequested(31),** **OCTET STRING(SIZE(0..63))**
 1567 OCTETS: The administratively defined coded character set name of the target queue
 1568 requested by the submitting user. For configuration 1, its value corresponds to the queue
 1569 in the device for which the agent is providing access. For configuration 2 and 3, its value
 1570 is the name of the queue that the user supplied to indicate to the server on which device(s)
 1571 they wanted the job to be processed.
 1572

1573 NOTE - typically an implementation SHOULD support either the **deviceNameRequested**
 1574 or **queueNameRequested** attribute, but not both.
 1575

1576 **physicalDevice(32),** **hrDeviceIndex**
 1577 AND/OR
 1578 **OCTET STRING(SIZE(0..63))**
 1579 INTEGER: MULTI-ROW: The index of the physical device MIB instance
 1580 requested/used, such as the Printer MIB[print-mib]. This value is an **hrDeviceIndex**
 1581 value. See the Host Resources MIB[hr-mib].
 1582
 1583 AND/OR

1584
 1585 OCTETS: MULTI-ROW: The name of the physical device to which the job is assigned.
 1586
 1587 **numberOfDocuments(33), Integer32(-2..2147483647)**
 1588 INTEGER: The number of documents in this job.
 1589
 1590 **fileName(34), OCTET STRING(SIZE(0..63))**
 1591 OCTETS: MULTI-ROW: The coded character set file name or URI[URI-spec] of the
 1592 document.
 1593
 1594 There is no restriction on the same file name occurring in multiple rows.
 1595
 1596 **documentName(35), OCTET STRING(SIZE(0..63))**
 1597 OCTETS: MULTI-ROW: The coded character set name of the document.
 1598
 1599 There is no restriction on the same document name occurring in multiple rows.
 1600
 1601 **jobComment(36), OCTET STRING(SIZE(0..63))**
 1602 OCTETS: An arbitrary human-readable coded character text string supplied by the
 1603 submitting user or the job submitting application program for any purpose. For example,
 1604 a user might indicate what he/she is going to do with the printed output or the job
 1605 submitting application program might indicate how the document was produced.
 1606
 1607 The **jobComment** attribute is not intended to be a name; see the **jobName** attribute.
 1608
 1609 **documentFormatIndex(37), Integer32(0..2147483647)**
 1610 INTEGER: MULTI-ROW: The index in the **prtInterpreterTable** in the Printer
 1611 MIB[print-mib] of the page description language (PDL) or control language interpreter
 1612 that this job requires/uses. A document or a job MAY use more than one PDL or control
 1613 language.
 1614
 1615 NOTE - As with all intensive attributes where multiple rows are allowed, there SHALL be
 1616 only one distinct row for each distinct interpreter; there SHALL be no duplicates.
 1617
 1618 NOTE - This attribute type is intended to be used with an agent that implements the
 1619 Printer MIB and SHALL not be used if the agent does not implement the Printer MIB.
 1620 Such an agent SHALL use the **documentFormat** attribute instead.
 1621
 1622 **documentFormat(38), PrtInterpreterLangFamilyTC**
 1623 **AND/OR**
 1624 **OCTET STRING(SIZE(0..63))**
 1625 INTEGER: MULTI-ROW: The interpreter language family corresponding to the Printer
 1626 MIB[print-mib] **prtInterpreterLangFamily** object, that this job requires/uses. A
 1627 document or a job MAY use more than one PDL or control language.
 1628
 1629 AND/OR
 1630

1631 OCTETS: MULTI-ROW: The document format registered as a media type[iana-media-
 1632 types], i.e., the name of the MIME content-type/subtype. Examples:
 1633 'application/postscript', 'application/vnd.hp-PCL', and 'application/pdf'
 1634
 1635

1636 ++++++
 1637 + **Job Parameter attributes**
 1638 +
 1639 + **The following attributes represent input parameters**
 1640 + **supplied by the submitting client in the job submission**
 1641 + **protocol.**
 1642 ++++++

1643
 1644 **jobPriority(50), Integer32(1..100)**
 1645 INTEGER: The priority for scheduling the job. It is used by servers and devices that
 1646 employ a priority-based scheduling algorithm.
 1647

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

1653
 1654 **jobProcessAfterDateAndTime(51), DateAndTime (SNMPv2-TC)**
 1655 OCTETS: The calendar date and time of day after which the job SHALL become a
 1656 candidate to be scheduled for processing. If the value of this attribute is in the future, the
 1657 server SHALL set the value of the job's **jmJobState** object to **pendingHeld** and add the
 1658 **jobProcessAfterSpecified** bit value to the job's **jmJobStateReasons1** object. When the
 1659 specified date and time arrives, the server SHALL remove the **jobProcessAfterSpecified**
 1660 bit value from the job's **jmJobStateReasons1** object and, if no other reasons remain,
 1661 SHALL change the job's **jmJobState** object to **pending**.
 1662

1663 **jobHold(52), JmBooleanTC**
 1664 INTEGER: If the value is 'true(4)', a client has explicitly specified that the job is to be
 1665 held until explicitly released. Until the job is explicitly released by a client, the job SHALL
 1666 be in the **pendingHeld** state with the **jobHoldSpecified** value in the
 1667 **jmJobStateReasons1** attribute.
 1668

1669 **jobHoldUntil(53), OCTET STRING(SIZE(0..63))**
 1670 OCTETS: The named time period during which the job SHALL become a candidate for
 1671 processing, such as 'evening', 'night', 'weekend', 'second-shift', 'third-shift', etc., as
 1672 defined by the system administrator. See IPP [ipp-model] for the standard keyword
 1673 values. Until that time period arrives, the job SHALL be in the **pendingHeld** state with
 1674 the **jobHoldUntilSpecified** value in the **jmJobStateReasons1** object. The value 'no-
 1675 hold' SHALL indicate explicitly that no time period has been specified.
 1676

1677 **outputBin(54), Integer32(0..2147483647)**
 1678 **AND/OR**

1679 **OCTET STRING(SIZE(0..63))**
1680 INTEGER: MULTI-ROW: The output subunit index in the Printer MIB[print-mib]
1681
1682 AND/OR
1683
1684 OCTETS: the name or number (represented as ASCII digits) of the output bin to which
1685 all or part of the job is placed in.
1686
1687 **sides(55), Integer32(-2..2)**
1688 INTEGER: MULTI-ROW: The number of sides, '1' or '2', that any document in this job
1689 requires/used.
1690
1691 **finishing(56), JmFinishingTC**
1692 INTEGER: MULTI-ROW: Type of finishing that any document in this job requires/used.
1693
1694
1695 ++++++
1696 + **Image Quality attributes (requested and consumed)**
1697 +
1698 + **For devices that can vary the image quality.**
1699 ++++++
1700
1701 **printQualityRequested(70), JmPrintQualityTC**
1702 INTEGER: MULTI-ROW: The print quality selection requested for a document in the
1703 job for printers that allow quality differentiation.
1704
1705 **printQualityUsed(71), JmPrintQualityTC**
1706 INTEGER: MULTI-ROW: The print quality selection actually used by a document in the
1707 job for printers that allow quality differentiation.
1708
1709 **printerResolutionRequested(72), JmPrinterResolutionTC**
1710 OCTETS: MULTI-ROW: The printer resolution requested for a document in the job for
1711 printers that support resolution selection.
1712
1713 **printerResolutionUsed(73), JmPrinterResolutionTC**
1714 OCTETS: MULTI-ROW: The printer resolution actually used by a document in the job
1715 for printers that support resolution selection.
1716
1717 **tonerEcomonyRequested(74), JmTonerEconomyTC**
1718 INTEGER: MULTI-ROW: The print quality selection requested for documents in the
1719 job for printers that allow toner quality differentiation.
1720
1721 **tonerEcomonyUsed(75), JmTonerEconomyTC**
1722 INTEGER: MULTI-ROW: The print quality selection actually used by documents in the
1723 job for printers that allow toner quality differentiation.
1724
1725 **tonerDensityRequested(76), Integer32(-2..100)**
1726 INTEGER: MULTI-ROW: The toner density requested for a document in this job for
1727 devices that can vary toner density levels. Level 1 is the lowest density and level 100 is

1728 the highest density level. Devices with a smaller range, SHALL map the 1-100 range
1729 evenly onto the implemented range.
1730

1731 **tonerDensityUsed(77), Integer32(-2..100)**

1732 INTEGER: MULTI-ROW: The toner density used by documents in this job for devices
1733 that can vary toner density levels. Level 1 is the lowest density and level 100 is the highest
1734 density level. Devices with a smaller range, SHALL map the 1-100 range evenly onto the
1735 implemented range.
1736
1737

1738 ++++++
1739 + **Job Progress attributes (requested and consumed)**

1740 +
1741 + **Pairs of these attributes can be used by monitoring**
1742 + **applications to show an indication of relative progress**
1743 + **to users.**
1744 ++++++

1745
1746 **jobCopiesRequested(90), Integer32(-2..2147483647)**

1747 INTEGER: The number of copies of the entire job that are to be produced.
1748

1749 **jobCopiesCompleted(91), Integer32(-2..2147483647)**

1750 INTEGER: The number of copies of the entire job that have been completed so far.
1751

1752 **documentCopiesRequested(92), Integer32(-2..2147483647)**

1753 INTEGER: The total count of the number of document copies requested. If there are
1754 documents A, B, and C, and document B is specified to produce 4 copies, the number of
1755 document copies requested is 6 for the job.
1756

1757 This attribute SHALL be used only when a job has multiple documents. The
1758 **jobCopiesRequested** attribute SHALL be used when the job has only one document.
1759

1760 **documentCopiesCompleted(93), Integer32(-2..2147483647)**

1761 INTEGER: The total count of the number of document copies completed so far for the
1762 job as a whole. If there are documents A, B, and C, and document B is specified to
1763 produce 4 copies, the number of document copies starts a 0 and runs up to 6 for the job as
1764 the job processes.
1765

1766 This attribute SHALL be used only when a job has multiple documents. The
1767 **jobCopiesCompleted** attribute SHALL be used when the job has only one document.
1768

1769 **jobKOctetsTransferred(94), Integer32(-2..2147483647)**

1770 INTEGER: The number of K (1024) octets transferred to the server or device to which
1771 the agent is providing access. This count is independent of the number of copies of the
1772 job or documents that will be produced, but it is only a measure of the number of bytes
1773 transferred to the server or device.
1774

1775 The agent SHALL round the actual number of octets transferred up to the next higher K.
1776 Thus 0 octets SHALL be represented as '0', 1-1024 octets SHALL BE represented as '1',

1777 1025-2048 SHALL be '2', etc. When the job completes, the values of the
1778 **jmJobKOctetsRequested** object and the **jobKOctetsTransferred** attribute SHALL be
1779 equal.
1780

1781 NOTE - The **jobKOctetsTransferred** can be used with the **jmJobKOctetsRequested**
1782 object in order to produce a relative indication of the progress of the job for agents that do
1783 not implement the **jmJobKOctetsProcessed** object.
1784

1785
1786 ++++++

1787 + **Impression attributes**

1788 +
1789 + **For a print job, an impression is the marking of the**
1790 + **entire side of a sheet. Two-sided processing involves two**
1791 + **impressions per sheet. Two-up is the placement of two**
1792 + **logical pages on one side of a sheet and so is still a**
1793 + **single impression. See also jmJobImpressionsRequested and**
1794 + **jmJobImpressionsCompleted objects in the jmJobTable.**

1795 ++++++

1796
1797 **impressionsSpooled(110), Integer32(-2..2147483647)**
1798 INTEGER: The number of impressions spooled to the server or device for the job so far.
1799

1800 **impressionsSentToDevice(111), Integer32(-2..2147483647)**
1801 INTEGER: The number of impressions sent to the device for the job so far.
1802

1803 **impressionsInterpreted(112), Integer32(-2..2147483647)**
1804 INTEGER: The number of impressions interpreted for the job so far.
1805

1806 **impressionsCompletedCurrentCopy(113), Integer32(-2..2147483647)**
1807 INTEGER: The number of impressions completed by the device for the current copy of
1808 the current document so far. For printing, the impressions completed includes
1809 interpreting, marking, and stacking the output. For other types of job services, the
1810 number of impressions completed includes the number of impressions processed.
1811

1812 This value SHALL be reset to **0** for each document in the job and for each document
1813 copy.
1814

1815 **fullColorImpressionsCompleted(114), Integer32(-2..2147483647)**
1816 INTEGER: The number of full color impressions completed by the device for this job so
1817 far. For printing, the impressions completed includes interpreting, marking, and stacking
1818 the output. For other types of job services, the number of impressions completed includes
1819 the number of impressions processed. Full color impressions are typically defined as those
1820 requiring 3 or more colorants, but this MAY vary by implementation.
1821

1822 **highlightColorImpressionsCompleted(115), Integer32(-2..**
1823 **2147483647)**
1824 INTEGER: The number of highlight color impressions completed by the device for this
1825 job so far. For printing, the impressions completed includes interpreting, marking, and

1826 stacking the output. For other types of job services, the number of impressions completed
1827 includes the number of impressions processed. Highlight color impressions are typically
1828 defined as those requiring black plus one other colorant, but this MAY vary by
1829 implementation.

1832 ++++++

1833 + Page attributes

1834 +
1835 + A page is a logical page. Number up can impose more than
1836 + one page on a single side of a sheet. Two-up is the
1837 + placement of two logical pages on one side of a sheet so
1838 + that each side counts as two pages.

1839 ++++++

1840
1841 pagesRequested(130), Integer32(-2..2147483647)
1842 INTEGER: The number of logical pages requested by the job to be processed.

1843
1844 pagesCompleted(131), Integer32(-2..2147483647)
1845 INTEGER: The number of logical pages completed for this job so far.

1846
1847 pagesCompletedCurrentCopy(132), Integer32(-2..2147483647)
1848 INTEGER: The number of logical pages completed for the current copy of the document
1849 so far. This value SHALL be reset to 0 for each document in the job and for each
1850 document copy.

1851
1852 ++++++

1853 + Sheet attributes

1854 +
1855 + The sheet is a single piece of a medium, whether printing
1856 + on one or both sides.

1857 ++++++

1858
1859 sheetsRequested(150), Integer32(-2..2147483647)
1860 INTEGER: The number of medium sheets requested to be processed for this job.

1861
1862 sheetsCompleted(151), Integer32(-2..2147483647)
1863 INTEGER: The number of medium sheets that have completed marking and stacking for
1864 the entire job so far whether those sheets have been processed on one side or on both.

1865
1866 sheetsCompletedCurrentCopy(152), Integer32(-2..2147483647)
1867 INTEGER: The number of medium sheets that have completed marking and stacking for
1868 the current copy of a document in the job so far whether those sheets have been processed
1869 on one side or on both.

1870
1871
1872 The value of this attribute SHALL be reset to 0 as each document in the job starts being
1873 processed and for each document copy as it starts being processed.

1874

1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923

++++
+ **Resources attributes (requested and consumed)**
+
+ **Pairs of these attributes can be used by monitoring**
+ **applications to show an indication of relative usage to**
+ **users.**
++++

mediumRequested(170), **JmMediumTypeTC**
AND/OR
OCTET STRING(SIZE(0..63))

INTEGER: MULTI-ROW: The type
AND/OR
OCTETS: the name of the medium that is required by the job.

mediumConsumed(171), **Integer32(-2..2147483647)**
AND
OCTET STRING(SIZE(0..63))

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

This attribute SHALL have both **Integer32** and **OCTET STRING** values.

colorantRequested(172), **Integer32(-2..2147483647)**
AND/OR
OCTET STRING(SIZE(0..63))

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

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

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

++++
+ **Time attributes (set by server or device)**
+
+ **This section of attributes are ones that are set by the**
+ **server or device that accepts jobs. Two forms of time are**
+ **provided. Each form is represented in a separate attribute.**

1924 + See section 3.1.2 and section 3.1.3 for the
 1925 + conformance requirements for time attribute for agents and
 1926 + monitoring applications, respectively. The two forms are:
 1927 +
 1928 + 'DateAndTime' is an 8 or 11 octet binary encoded year,
 1929 + month, day, hour, minute, second, deci-second with
 1930 + optional offset from UTC. See SNMPv2-TC [SMIv2-TC].
 1931 +
 1932 + NOTE: 'DateAndTime' is not printable characters; it is
 1933 + binary.
 1934 +
 1935 + 'JmTimeStampTC' is the time of day measured in the number of
 1936 + seconds since the system was booted.
 1937 ++++++

1939 **jobSubmissionToServerTime(190),** **JmTimeStampTC**
 1940 **AND/OR**
 1941 **DateAndTime**
 1942 INTEGER: Configuration 3 only: The time
 1943 AND/OR
 1944 OCTETS: the date and time that the job was submitted to the server (as distinguished
 1945 from the device which uses jobSubmissionTime).
 1946

1947 **jobSubmissionTime(191),** **JmTimeStampTC**
 1948 **AND/OR**
 1949 **DateAndTime**
 1950 INTEGER: Configurations 1, 2, and 3: The time
 1951 AND/OR
 1952 OCTETS: the date and time that the job was submitted to the server or device to which
 1953 the agent is providing access.
 1954
 1955

1956 **jobStartedBeingHeldTime(192),** **JmTimeStampTC**
 1957 **AND/OR**
 1958 **DateAndTime**
 1959 INTEGER: The time
 1960 AND/OR
 1961 OCTETS: the date and time that the job last entered the **pendingHeld** state. If the job
 1962 has never entered the **pendingHeld** state, then the value SHALL be '0' or the attribute
 1963 SHALL not be present in the table.
 1964
 1965

1966 **jobStartedProcessingTime(193),** **JmTimeStampTC**
 1967 **AND/OR**
 1968 **DateAndTime**
 1969 INTEGER: The time
 1970 AND/OR
 1971 OCTETS: the date and time that the job started processing.
 1972

1973 **jobCompletedTime(194),** **JmTimeStampTC**
 1974 **AND/OR**
 1975 **DateAndTime**
 1976 INTEGER: The time
 1977 AND/OR
 1978 OCTETS: the date and time that the job entered the **completed, canceled, or aborted**
 1979 state.
 1980
 1981 **jobProcessingCPUtime(195)** **Integer32(-2..2147483647)**
 1982 **UNITS 'seconds'**
 1983 INTEGER: The amount of CPU time in seconds that the job has been in the **processing**
 1984 state. If the job enters the **processingStopped** state, that elapsed time SHALL not be
 1985 included. In other words, the **jobProcessingCPUtime** value SHOULD be relatively
 1986 repeatable when the same job is processed again on the same device."
 1987
 1988 REFERENCE
 1989 "See Section 3.2 entitled 'The Attribute Mechanism' for a description of this textual-convention
 1990 and its use in the **jmAttributeTable**.
 1991
 1992 This is a type 2 enumeration. See Section 3.6.1.2."
 1993 SYNTAX INTEGER {
 1994 other(1),
 1995 unknown(2),
 1996 jobStateReasons2(3),
 1997 jobStateReasons3(4),
 1998 jobStateReasons4(5),
 1999 processingMessage(6),
 2000
 2001 jobAccountName(21),
 2002 serverAssignedJobName(22),
 2003 jobName(23),
 2004 jobServiceTypes(24),
 2005 jobSourceChannelIndex(25),
 2006 jobSourcePlatformType(26),
 2007 submittingServerName(27),
 2008 submittingApplicationName(28),
 2009 jobOriginatingHost(29),
 2010 deviceNameRequested(30),
 2011 queueNameRequested(31),
 2012 physicalDevice(32),
 2013 numberOfDocuments(33),
 2014 fileName(34),
 2015 documentName(35),
 2016 jobComment(36),
 2017 documentFormatIndex(37),
 2018 documentFormat(38),
 2019
 2020 jobPriority(50),
 2021 jobProcessAfterDateAndTime(51),

```
2022     jobHold(52),
2023     jobHoldUntil(53),
2024     outputBin(54),
2025     sides(55),
2026     finishing(56),
2027
2028     printQualityRequested(70),
2029     printQualityUsed(71),
2030     printerResolutionRequested(72),
2031     printerResolutionUsed(73),
2032     tonerEcomonyRequested(74),
2033     tonerEcomonyUsed(75),
2034     tonerDensityRequested(76),
2035     tonerDensityUsed(77),
2036
2037     jobCopiesRequested(90),
2038     jobCopiesCompleted(91),
2039     documentCopiesRequested(92),
2040     documentCopiesCompleted(93),
2041     jobKOctetsTransferred(94),
2042
2043     impressionsSpooled(110),
2044     impressionsSentToDevice(111),
2045     impressionsInterpreted(112),
2046     impressionsCompletedCurrentCopy(113),
2047     fullColorImpressionsCompleted(114),
2048     highlightColorImpressionsCompleted(115),
2049
2050     pagesRequested(130),
2051     pagesCompleted(131),
2052     pagesCompletedCurrentCopy(132),
2053
2054     sheetsRequested(150),
2055     sheetsCompleted(151),
2056     sheetsCompletedCurrentCopy(152),
2057
2058     mediumRequested(170),
2059     mediumConsumed(171),
2060     colorantRequested(172),
2061     colorantConsumed(173),
2062
2063     jobSubmissionToServerTime(190),
2064     jobSubmissionTime(191),
2065     jobStartedBeingHeldTime(192),
2066     jobStartedProcessingTime(193),
2067     jobCompletedTime(194),
2068     jobProcessingCPUTime(195)
2069 }
2070
```

2071

2072

2073

2074 **JmJobServiceTypesTC ::= TEXTUAL-CONVENTION**2075 **STATUS** current2076 **DESCRIPTION**

2077 "Specifies the type(s) of service to which the job has been submitted (print, fax, scan, etc.). The
 2078 service type is represented as an enum that is bit encoded with each job service type so that
 2079 more general and arbitrary services can be created, such as services with more than one
 2080 destination type, or ones with only a source or only a destination. For example, a job service
 2081 might **scan**, **faxOut**, and **print** a single job. In this case, three bits would be set in the
 2082 **jobServiceTypes** attribute, corresponding to the hexadecimal values: **0x8** + **0x20** + **0x4**,
 2083 respectively, yielding: **0x2C**.

2084

2085 Whether this attribute is set from a job attribute supplied by the job submission client or is set by
 2086 the recipient job submission server or device depends on the job submission protocol. With
 2087 either implementation, the agent SHALL return a non-zero value for this attribute indicating the
 2088 type of the job.

2089

2090 One of the purposes of this attribute is to permit a requester to filter out jobs that are not of
 2091 interest. For example, a printer operator MAY only be interested in jobs that include printing.
 2092 That is why the attribute is in the job identification category.

2093

2094 The following service component types are defined (in hexadecimal) and are assigned a separate
 2095 bit value for use with the **jobServiceTypes** attribute:

2096

2097

other 0x1

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

2098

2099

unknown**0x2**

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

2100

2101

2102

print 0x4

The job contains some instructions that specify printing

2103

2104

scan 0x8

The job contains some instructions that specify scanning

2105

2106

faxIn 0x10

The job contains some instructions that specify receive fax

2107

2108

faxOut**0x20**

The job contains some instructions that specify sending fax

2109

2110

getFile**0x40**

The job contains some instructions that specify accessing files or documents

2111

2112

putFile**0x80**

2113

2114

2115

2116

2117

2118

2164	jobIncoming	0x4
2165	The job has been accepted by the server or device, but the server or device is expecting	
2166	(1) additional operations from the client to finish creating the job and/or (2) is	
2167	accessing/accepting document data.	
2168		
2169	jobOutgoing	0x8
2170	Configuration 2 only: The server is transmitting the job to the device.	
2171		
2172	jobHoldSpecified	0x10
2173	The value of the job's jobHold(52) attribute is TRUE. The job SHALL NOT be a	
2174	candidate for processing until this reason is removed and there are no other reasons to	
2175	hold the job.	
2176		
2177	jobHoldUntilSpecified	0x20
2178	The value of the job's jobHoldUntil(53) attribute specifies a time period that is still in the	
2179	future. The job SHALL NOT be a candidate for processing until this reason is removed	
2180	and there are no other reasons to hold the job.	
2181		
2182	jobProcessAfterSpecified	0x40
2183	The value of the job's jobProcessAfterDateAndTime(51) attribute specifies a time that is	
2184	still in the future. The job SHALL NOT be a candidate for processing until this reason is	
2185	removed and there are no other reasons to hold the job.	
2186		
2187	resourcesAreNotReady	0x80
2188	At least one of the resources needed by the job, such as media, fonts, resource objects,	
2189	etc., is not ready on any of the physical devices for which the job is a candidate. This	
2190	condition MAY be detected when the job is accepted, or subsequently while the job is	
2191	pending or processing , depending on implementation.	
2192		
2193	deviceStoppedPartly	0x100
2194	One or more, but not all, of the devices to which the job is assigned are stopped. If all of	
2195	the devices are stopped (or the only device is stopped), the deviceStopped reason	
2196	SHALL be used.	
2197		
2198	deviceStopped	0x200
2199	The device(s) to which the job is assigned is (are all) stopped.	
2200		
2201	jobPrinting	0x400
2202	The output device is marking media. This attribute is useful for servers and output devices	
2203	which spend a great deal of time processing when no marking is happening and then want	
2204	to show that marking is now happening or when the job is in the canceled or aborted	
2205	state, but the marking has not yet stopped so that impression or sheet counts are still	
2206	increasing for the job.	
2207		
2208	jobCanceledByUser	0x800
2209	The job was canceled by the user, i.e., by an unknown user or by a user whose name is the	
2210	same as the value of the job's jmJobOwner object.	
2211		

2212	jobCanceledByOperator	0x1000
2213	The job was canceled by the operator, i.e., by a user whose name is different than the	
2214	value of the job's jmJobOwner object.	
2215		
2216	abortedBySystem	0x2000
2217	The job was aborted by the system.	
2218		
2219	NOTE - When the system puts a job into the 'aborted' job state, this reason is not needed.	
2220	This reason is needed only when the system aborts a job, but, instead of placing the job in	
2221	the aborted job state, places the job in the pendingHeld state, so that a user or operator	
2222	can manually try the job again.	
2223		
2224	jobCompletedSuccessfully	0x4000
2225	The job completed successfully.	
2226		
2227	jobCompletedWithWarnings	0x8000
2228	The job completed with warnings.	
2229		
2230	jobCompletedWithErrors	0x10000
2231	The job completed with errors (and possibly warnings too).	
2232		
2233	The following additional job state reasons have been added to represent job states that are	
2234	in ISO DPA[iso-dpa] and other job submission protocols:	
2235		
2236	jobPaused	0x20000
2237	The job has been indefinitely suspended by a client issuing an operation to suspend the job	
2238	so that other jobs may proceed using the same devices. The client MAY issue an	
2239	operation to resume the paused job at any time, in which case the agent SHALL remove	
2240	the jobPaused values from the job's jmJobStateReasons1 object and the job is eventually	
2241	resumed at or near the point where the job was paused.	
2242		
2243	jobInterrupted	0x40000
2244	The job has been interrupted while processing by a client issuing an operation that	
2245	specifies another job to be run instead of the current job. The server or device will	
2246	automatically resume the interrupted job when the interrupting job completes.	
2247		
2248	jobRetained	0x80000
2249	The job is being retained by the server or device with all of the job's document data (and	
2250	submitted resources, such as fonts, logos, and forms, if any). Thus a client could issue an	
2251	operation to the server or device to either (1) re-do the job (or a copy of the job) on the	
2252	same server or device or (2) resubmit the job to another server or device. When a client	
2253	could no longer re-do/resubmit the job, such as after the document data has been	
2254	discarded, the agent SHALL remove the jobRetained value from the	
2255	jmJobStateReasons1 object."	
2256	REFERENCE	
2257	"These bit definitions are the equivalent of a type 2 enum except that combinations of bits may	
2258	be used together. See section 3.6.1.2. The remaining bits are reserved for future	
2259	standardization and/or registration."	
2260		

2261 SYNTAX INTEGER(0..2147483647) -- 31 bits, all but sign bit

2262
2263
2264
2265
2266

2267 **JmJobStateReasons2TC** ::= TEXTUAL-CONVENTION

2268 STATUS current

2269 DESCRIPTION

2270 "This textual-convention is used with the **jobStateReasons2** attribute to provides additional
2271 information regarding the **jmJobState** object. See the description under
2272 **JmJobStateReasons1TC** for additional information that applies to all reasons.

2273

2274 The following standard values are defined (in hexadecimal) as *powers of two*, since multiple
2275 values may be used at the same time:

2276

2277 **cascaded** **0x1**

2278 An outbound gateway has transmitted all of the job's job and document attributes and data
2279 to another spooling system.

2280

2281 **deletedByAdministrator** **0x2**

2282 The administrator has deleted the job.

2283

2284 **discardTimeArrived** **0x4**

2285 The job has been deleted due to the fact that the time specified by the job's job-discard-
2286 time attribute has arrived.

2287

2288 **postProcessingFailed** **0x8**

2289 The post-processing agent failed while trying to log accounting attributes for the job;
2290 therefore the job has been placed into the completed state with the **jobRetained**
2291 **jmJobStateReasons1** object value for a system-defined period of time, so the
2292 administrator can examine it, resubmit it, etc.

2293

2294 **submissionInterrupted** **0x10**

2295 Indicates that the job was not completely submitted for some unforeseen reason, such as:
2296 (1) the server has crashed before the job was closed by the client, (2) the server or the
2297 document transfer method has crashed in some non-recoverable way before the document
2298 data was entirely transferred to the server, (3) the client crashed or failed to close the job
2299 before the time-out period.

2300

2301 **maxJobFaultCountExceeded** **0x20**

2302 The job has faulted several times and has exceeded the administratively defined fault count
2303 limit.

2304

2305 **devicesNeedAttentionTimeOut** **0x40**

2306 One or more document transforms that the job is using needs human intervention in order
2307 for the job to make progress, but the human intervention did not occur within the site-
2308 settable time-out value.

2309

2310	needsKeyOperatorTimeOut	0x80
2311	One or more devices or document transforms that the job is using need a specially trained	
2312	operator (who may need a key to unlock the device and gain access) in order for the job to	
2313	make progress, but the key operator intervention did not occur within the site-settable	
2314	time-out value.	
2315		
2316	jobStartWaitTimeOut	0x100
2317	The server/device has stopped the job at the beginning of processing to await human	
2318	action, such as installing a special cartridge or special non-standard media, but the job was	
2319	not resumed within the site-settable time-out value and the server/device has transitioned	
2320	the job to the pendingHeld state.	
2321		
2322	jobEndWaitTimeOut	0x200
2323	The server/device has stopped the job at the end of processing to await human action,	
2324	such as removing a special cartridge or restoring standard media, but the job was not	
2325	resumed within the site-settable time-out value and the server/device has transitioned the	
2326	job to the completed state.	
2327		
2328	jobPasswordWaitTimeOut	0x400
2329	The server/device has stopped the job at the beginning of processing to await input of the	
2330	job's password, but the password was not received within the site-settable time-out value.	
2331		
2332	deviceTimedOut	0x800
2333	A device that the job was using has not responded in a period specified by the device's	
2334	site-settable attribute.	
2335		
2336	connectingToDeviceTimeOut	0x1000
2337	The server is attempting to connect to one or more devices which may be dial-up, polled,	
2338	or queued, and so may be busy with traffic from other systems, but server was unable to	
2339	connect to the device within the site-settable time-out value.	
2340		
2341	transferring	0x2000
2342	The job is being transferred to a down stream server or device.	
2343		
2344	queuedInDevice	0x4000
2345	The job has been queued in a down stream server or device.	
2346		
2347	jobCleanup	0x8000
2348	The server/device is performing cleanup activity as part of ending normal processing.	
2349		
2350	processingToStopPoint	0x10000
2351	The requester has issued an operation to interrupt the job and the server/device is	
2352	processing up until the specified stop point occurs.	
2353		
2354	jobPasswordWait	0x20000
2355	The server/device has selected the job to be next to process, but instead of assigning	
2356	resources and starting the job processing, the server/device has transitioned the job to the	
2357	pendingHeld state to await entry of a password (and dispatched another job, if there is	
2358	one).	

2359		
2360	validating	0x40000
2361	The server/device is validating the job <i>after</i> accepting the job.	
2362		
2363	queueHeld	0x80000
2364	The operator has held the entire job set or queue.	
2365		
2366	jobProofWait	0x100000
2367	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.	
2368		
2369		
2370		
2371	heldForDiagnostics	0x200000
2372	The system is running intrusive diagnostics, so that all jobs are being held.	
2373		
2374	serviceOffLine	0x400000
2375	The service/document transform is off-line and accepting no jobs. All pending jobs are put into the pendingHeld state. This could be true if its input is impaired or broken.	
2376		
2377		
2378	noSpaceOnServer	0x800000
2379	There is no room on the server to store all of the job.	
2380		
2381	pinRequired	0x1000000
2382	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.	
2383		
2384		
2385	exceededAccountLimit	0x2000000
2386	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.	
2387		
2388		
2389		
2390		
2391	heldForRetry	0x4000000
2392	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 inaccessible. For such a situation, the server/device SHALL transition the job from the processing to the pendingHeld , rather than to the aborted state.	
2393		
2394		
2395		
2396		
2397		
2398	The following values are from the X/Open PSIS draft standard:	
2399		
2400	canceledByShutdown	0x8000000
2401	The job was canceled because the server or device was shutdown before completing the job.	
2402		
2403		
2404	deviceUnavailable	0x10000000
2405	This job was aborted by the system because the device is currently unable to accept jobs.	
2406		

2407 **wrongDevice** **0x20000000**
2408 This job was aborted by the system because the device is unable to handle this particular
2409 job; the spooler SHOULD try another device or the user should submit the job to another
2410 device.
2411
2412 **badJob** **0x40000000**
2413 This job was aborted by the system because this job has a major problem, such as an ill-
2414 formed PDL; the spooler SHOULD not even try another device. "
2415 REFERENCE
2416 "These bit definitions are the equivalent of a type 2 enum except that combinations of them may
2417 be used together. See section 3.6.1.2. See the description under **JmJobStateReasons1TC** and
2418 the **jobStateReasons2** attribute."
2419
2420 SYNTAX **INTEGER(0..2147483647)** -- 31 bits, all but sign bit
2421
2422
2423
2424
2425
2426
2427 **JmJobStateReasons3TC** ::= TEXTUAL-CONVENTION
2428 STATUS current
2429 DESCRIPTION
2430 "This textual-convention is used with the **jobStateReasons3** attribute to provides additional
2431 information regarding the **jmJobState** object. See the description under
2432 **JmJobStateReasons1TC** for additional information that applies to all reasons.
2433
2434 The following standard values are defined (in hexadecimal) as *powers of two*, since multiple
2435 values may be used at the same time:
2436
2437 **jobInterruptedByDeviceFailure** **0x1**
2438 A device or the print system software that the job was using has failed while the job was
2439 processing. The server or device is keeping the job in the **pendingHeld** state until an
2440 operator can determine what to do with the job."
2441 REFERENCE
2442 "These bit definitions are the equivalent of a type 2 enum except that combinations of them may
2443 be used together. See section 3.6.1.2. The remaining bits are reserved for future
2444 standardization and/or registration. See the description under **JmJobStateReasons1TC** and the
2445 **jobStateReasons3** attribute."
2446 SYNTAX **INTEGER(0..2147483647)** -- 31 bits, all but sign bit
2447
2448
2449
2450
2451 **JmJobStateReasons4TC** ::= TEXTUAL-CONVENTION
2452 STATUS current
2453 DESCRIPTION
2454

2455 "This textual-convention is used in the **jobStateReasons4** attribute to provides additional
2456 information regarding the **jmJobState** object. See the description under
2457 **JmJobStateReasons1TC** for additional information that applies to all reasons.
2458

2459 The following standard values are defined (in hexadecimal) as *powers of two*, since multiple
2460 values may be used at the same time:
2461

2462 none yet defined. These bits are reserved for future standardization and/or registration."
2463

2463 REFERENCE

2464 "These bit definitions are the equivalent of a type 2 enum except that combinations of them may
2465 be used together. See section 3.6.1.2. See the description under **JmJobStateReasons1TC** and
2466 the **jobStateReasons4** attribute."
2467

2468 SYNTAX **INTEGER(0..2147483647)** -- 31 bits, all but sign bit

```

2469
2470 jobmonMIBObjects OBJECT IDENTIFIER ::= { jobmonMIB 1 }
2471
2472 -- The General Group (MANDATORY)
2473
2474 -- The jmGeneralGroup consists entirely of the jmGeneralTable.
2475
2476 jmGeneral OBJECT IDENTIFIER ::= { jobmonMIBObjects 1 }
2477
2478 jmGeneralTable OBJECT-TYPE
2479     SYNTAX     SEQUENCE OF JmGeneralEntry
2480     MAX-ACCESS not-accessible
2481     STATUS     current
2482     DESCRIPTION
2483         "The jmGeneralTable consists of information of a general nature that are per-job-set, but are
2484         not per-job. See Section 2 entitled 'Terminology and Job Model' for the definition of a job set."
2485     REFERENCE
2486         "The MANDATORY-GROUP macro specifies that this group is MANDATORY."
2487     ::= { jmGeneral 1 }
2488
2489 jmGeneralEntry OBJECT-TYPE
2490     SYNTAX     JmGeneralEntry
2491     MAX-ACCESS not-accessible
2492     STATUS     current
2493     DESCRIPTION
2494         "Information about a job set (queue).
2495
2496         An entry SHALL exist in this table for each job set."
2497     INDEX { jmGeneralJobSetIndex }
2498     ::= { jmGeneralTable 1 }
2499
2500 JmGeneralEntry ::= SEQUENCE {
2501     jmGeneralJobSetIndex           Integer32(1..32767),
2502     jmGeneralNumberOfActiveJobs   Integer32(0..2147483647),
2503     jmGeneralOldestActiveJobIndex Integer32(0..2147483647),
2504     jmGeneralNewestActiveJobIndex Integer32(0..2147483647),
2505     jmGeneralJobPersistence       Integer32(15..2147483647),
2506     jmGeneralAttributePersistence Integer32(15..2147483647),
2507     jmGeneralJobSetName          OCTET STRING(SIZE(0..63))
2508 }
2509
2510 jmGeneralJobSetIndex OBJECT-TYPE
2511     SYNTAX     Integer32(1..32767)
2512     MAX-ACCESS not-accessible
2513     STATUS     current
2514     DESCRIPTION
2515         "A unique value for each job set in this MIB. The jmJobTable and jmAttributeTable tables
2516         have this same index as their primary index.
2517

```

2518 The value(s) of the **jmGeneralJobSetIndex** SHALL be persistent across power cycles, so that
 2519 clients that have retained **jmGeneralJobSetIndex** values will access the same job sets upon
 2520 subsequent power-up.
 2521

2522 An implementation that has only one job set, such as a printer with a single queue, SHALL hard
 2523 code this object with the value **1**."

2524 REFERENCE
 2525 "See Section 2 entitled 'Terminology and Job Model' for the definition of a job set.
 2526 Corresponds to the first index in **jmJobTable** and **jmAttributeTable**."
 2527 ::= { jmGeneralEntry 1 }
 2528

2529 **jmGeneralNumberOfActiveJobs** OBJECT-TYPE
 2530 SYNTAX Integer32(0..2147483647)
 2531 MAX-ACCESS read-only
 2532 STATUS current
 2533 DESCRIPTION
 2534 "The current number of 'active' jobs in the **jmJobIDTable**, **jmJobTable**, and
 2535 **jmAttributeTable**, i.e., the total number of jobs that are in the **pending**, **processing**, or
 2536 **processingStopped** states. See the **JmJobStateTC** textual-convention for the exact
 2537 specification of the semantics of the job states."
 2538 ::= { jmGeneralEntry 2 }
 2539

2540 **jmGeneralOldestActiveJobIndex** OBJECT-TYPE
 2541 SYNTAX Integer32 (0..2147483647)
 2542 MAX-ACCESS read-only
 2543 STATUS current
 2544 DESCRIPTION
 2545 "The **jmJobIndex** of the oldest job that is still in one of the 'active' states (**pending**, **processing**,
 2546 or **processingStopped**). In other words, the index of the 'active' job that has been in the job
 2547 tables the longest.
 2548
 2549 If there are no active jobs, the agent SHALL set the value of this object to **0**."
 2550 REFERENCE
 2551 "See Section 3.2 entitled 'The Job Tables and the Oldest Active and Newest Active Indexes' for
 2552 a description of the usage of this object."
 2553 ::= { jmGeneralEntry 3 }
 2554

2555 **jmGeneralNewestActiveJobIndex** OBJECT-TYPE
 2556 SYNTAX Integer32 (0..2147483647)
 2557 MAX-ACCESS read-only
 2558 STATUS current
 2559 DESCRIPTION
 2560 "The **jmJobIndex** of the newest job that is in one of the 'active' states (**pending**, **processing**, or
 2561 **processingStopped**). In other words, the index of the 'active' job that has been most recently
 2562 added to the **job tables**.
 2563
 2564 When all jobs become 'inactive', i.e., enter the **pendingHeld**, **completed**, **canceled**, or **aborted**
 2565 states, the agent SHALL set the value of this object to **0**."
 2566 REFERENCE

2567 "See Section 3.2 entitled 'The Job Tables and the Oldest Active and Newest Active Indexes' for
 2568 a description of the usage of this object."
 2569 ::= { jmGeneralEntry 4 }
 2570

jmGeneralJobPersistence OBJECT-TYPE
 2571 SYNTAX **Integer32(15..2147483647)**
 2572 UNITS "seconds"
 2573 MAX-ACCESS read-only
 2574 STATUS current
 2575 DESCRIPTION
 2576 "The minimum time in seconds for this instance of the Job Set that an entry SHALL remain in
 2577 the **jmJobIDTable** and **jmJobTable** after **processing** has *completed*, i.e., the minimum time in
 2578 seconds starting when the job enters the **completed, canceled, or aborted** state.
 2579
 2580 Depending on implementation, the value of this object MAY be either: (1) set by the system
 2581 administrator by means outside this specification or (2) fixed by the implementation.
 2582
 2583 This value SHALL be equal to or greater than the value of **jmGeneralAttributePersistence**.
 2584 This value SHOULD be at least 60 which gives a monitoring application one minute in which to
 2585 poll for job data."
 2586 DEFVAL { 60 } -- one minute
 2587 ::= { jmGeneralEntry 5 }
 2588
 2589

jmGeneralAttributePersistence OBJECT-TYPE
 2590 SYNTAX **Integer32(15..2147483647)**
 2591 UNITS "seconds"
 2592 MAX-ACCESS read-only
 2593 STATUS current
 2594 DESCRIPTION
 2595 "The minimum time in seconds for this instance of the Job Set that an entry SHALL remain in
 2596 the **jmAttributeTable** after **processing** has *completed* , i.e., the time in seconds starting when
 2597 the job enters the **completed, canceled, or aborted** state.
 2598
 2599 Depending on implementation, the value of this object MAY be either (1) set by the system
 2600 administrator by means outside this specification or MAY be (2) fixed by the implementation.
 2601
 2602 This value SHOULD be at least 60 which gives a monitoring application one minute in which to
 2603 poll for job data."
 2604 DEFVAL { 60 } -- one minute
 2605 ::= { jmGeneralEntry 6 }
 2606
 2607

jmGeneralJobSetName OBJECT-TYPE
 2608 SYNTAX **OCTET STRING(SIZE(0..63))**
 2609 MAX-ACCESS read-only
 2610 STATUS current
 2611 DESCRIPTION
 2612 "The human readable name of this job set assigned by the system administrator (by means
 2613 outside of this MIB). Typically, this name SHOULD be the name of the job queue. If a server
 2614 or device has only a single job set, this object can be the administratively assigned name of the
 2615

2616 server or device itself. This name does not need to be unique, though each job set in a single
2617 Job Monitoring MIB SHOULD have distinct names.
2618

2619 NOTE - The purpose of this object is to help the user of the job monitoring application
2620 distinguish between several job sets in implementations that support more than one job set."
2621 REFERENCE
2622 "See the OBJECT compliance macro for the minimum maximum length required for
2623 conformance."
2624 ::= { jmGeneralEntry 7 }
2625
2626
2627
2628
2629

2630 -- The Job ID Group (MANDATORY)
2631
2632 -- The **jmJobIDGroup** consists entirely of the **jmJobIDTable**.
2633

2634 jmJobID OBJECT IDENTIFIER ::= { jobmonMIBObjects 2 }
2635

2636 jmJobIDTable OBJECT-TYPE
2637 SYNTAX SEQUENCE OF JmJobIDEntry
2638 MAX-ACCESS not-accessible
2639 STATUS current
2640 DESCRIPTION
2641 "The **jmJobIDTable** provides a correspondence map (1) between the job submission ID that a
2642 client uses to refer to a job and (2) the **jmGeneralJobSetIndex** and **jmJobIndex** that the Job
2643 Monitoring MIB agent assigned to the job and that are used to access the job in all of the other
2644 tables in the MIB. If a monitoring application already knows the **jmGeneralJobSetIndex** and
2645 the **jmJobIndex** of the job it is querying, that application NEED NOT use the **jmJobIDTable**."
2646 REFERENCE
2647 "The MANDATORY-GROUP macro specifies that this group is MANDATORY."
2648 ::= { jmJobID 1 }
2649

2650 jmJobIDEntry OBJECT-TYPE
2651 SYNTAX JmJobIDEntry
2652 MAX-ACCESS not-accessible
2653 STATUS current
2654 DESCRIPTION
2655 "The map from (1) the **jmJobSubmissionID** to (2) the **jmGeneralJobSetIndex** and
2656 **jmJobIndex**.
2657
2658 An entry SHALL exist in this table for each job currently known to the agent for all job sets and
2659 job states. Each job SHALL appear in one and only one job set."
2660 INDEX { **jmJobSubmissionID** }
2661 ::= { jmJobIDTable 1 }
2662

2663 JmJobIDEntry ::= SEQUENCE {
2664 **jmJobSubmissionID** OCTET STRING(SIZE(48)),

```

2665     jmJobIDJobSetIndex                               Integer32(1..32767),
2666     jmJobIDJobIndex                                   Integer32(1..2147483647)
2667 }
2668
2669 jmJobSubmissionID OBJECT-TYPE
2670     SYNTAX      OCTET STRING(SIZE(48))
2671     MAX-ACCESS  not-accessible
2672     STATUS      current
2673     DESCRIPTION
2674         "A quasi-unique 48-octet fixed-length string ID which identifies the job within a particular
2675         client-server environment. There are multiple formats for the jmJobSubmissionID. See the
2676         JmJobSubmissionIDTypeTC textual convention. Each format SHALL be registered using the
2677         procedures of a type 2 enum. See section 3.6.3 entitled: 'IANA Registration of Job Submission
2678         Id Formats'.
2679
2680         If the requester (client or server) does not supply a job submission ID in the job submission
2681         protocol, then the recipient (server or device) SHALL assign a job submission ID using any of
2682         the standard formats and adding the final 8 octets to distinguish the ID from others submitted
2683         from the same requester.
2684
2685         The monitoring application, whether in the client or running separately, MAY use the job
2686         submission ID to help identify which jmJobIndex was assigned by the agent, i.e., in which row
2687         the job information is in the other tables.
2688
2689         NOTE - fixed-length is used so that a management application can use a shortened GetNext
2690         varbind (in SNMPv1 and SNMPv2) in order to get the next submission ID, disregarding the
2691         remainder of the ID in order to access jobs independent of the trailing identifier part, e.g., to get
2692         all jobs submitted by a particular jmJobOwner or from a particular MAC address."
2693     ::= { jmJobIDEntry 1 }
2694
2695 jmJobIDJobSetIndex OBJECT-TYPE
2696     SYNTAX      Integer32(1..32767)
2697     MAX-ACCESS  read-only
2698     STATUS      current
2699     DESCRIPTION
2700         "This object contains the value of the jmGeneralJobSetIndex for the job with the
2701         jmJobSubmissionID value, i.e., the job set index of the job set in which the job was placed
2702         when that server or device accepted the job. This 16-bit value in combination with the
2703         jmJobIDJobIndex value permits the management application to access the other tables to
2704         obtain the job-specific objects for this job."
2705     REFERENCE
2706         "See jmGeneralJobSetIndex in the jmGeneralTable."
2707     ::= { jmJobIDEntry 2 }
2708
2709 jmJobIDJobIndex OBJECT-TYPE
2710     SYNTAX      Integer32(1..2147483647)
2711     MAX-ACCESS  read-only
2712     STATUS      current
2713     DESCRIPTION

```

```

2714         "This object contains the value of the jmJobIndex for the job with the jmJobSubmissionID
2715         value, i.e., the job index for the job when the server or device accepted the job. This value, in
2716         combination with the jmJobIDJobSetIndex value, permits the management application to
2717         access the other tables to obtain the job-specific objects for this job."
2718     REFERENCE
2719         "See jmJobIndex in the jmJobTable."
2720     ::= { jmJobIDEntry 3 }
2721
2722
2723
2724
2725 -- The Job Group (MANDATORY)
2726
2727 -- The jmJobGroup consists entirely of the jmJobTable.
2728
2729 jmJob OBJECT IDENTIFIER ::= { jobmonMIBObjects 3 }
2730
2731 jmJobTable OBJECT-TYPE
2732     SYNTAX     SEQUENCE OF JmJobEntry
2733     MAX-ACCESS not-accessible
2734     STATUS     current
2735     DESCRIPTION
2736         "The jmJobTable consists of basic job state and status information for each job in a job set that
2737         (1) monitoring applications need to be able to access in a single SNMP Get operation, (2) that
2738         have a single value per job, and (3) that SHALL always be implemented."
2739     REFERENCE
2740         "The MANDATORY-GROUP macro specifies that this group is MANDATORY."
2741     ::= { jmJob 1 }
2742
2743 jmJobEntry OBJECT-TYPE
2744     SYNTAX     JmJobEntry
2745     MAX-ACCESS not-accessible
2746     STATUS     current
2747     DESCRIPTION
2748         "Basic per-job state and status information.
2749
2750         An entry SHALL exist in this table for each job, no matter what the state of the job is. Each job
2751         SHALL appear in one and only one job set."
2752     REFERENCE
2753         "See Section 3.2 entitled 'The Job Tables'."
2754     INDEX { jmGeneralJobSetIndex, jmJobIndex }
2755     ::= { jmJobTable 1 }
2756
2757 JmJobEntry ::= SEQUENCE {
2758     jmJobIndex                Integer32(1..2147483647),
2759     jmJobState                JmJobStateTC,
2760     jmJobStateReasons1       JmJobStateReasons1TC,
2761     jmNumberOfInterveningJobs Integer32(-2..2147483647),
2762     jmJobKOctetsRequested    Integer32(-2..2147483647),

```

```

2763     jmJobKOctetsProcessed                Integer32(-2..2147483647),
2764     jmJobImpressionsRequested          Integer32(-2..2147483647),
2765     jmJobImpressionsCompleted         Integer32(-2..2147483647),
2766     jmJobOwner                        OCTET STRING(SIZE(0..63))
2767 }
2768
2769 jmJobIndex OBJECT-TYPE
2770     SYNTAX      Integer32(1..2147483647)
2771     MAX-ACCESS  not-accessible
2772     STATUS      current
2773     DESCRIPTION
2774         "The sequential, monotonically increasing identifier index for the job generated by the server or
2775         device when that server or device accepted the job. This index value permits the management
2776         application to access the other tables to obtain the job-specific row entries.
2777
2778         Agents providing access to systems that contain jobs with a job identifier of 0 SHALL map the
2779         job identifier value 0 to a jmJobIndex value that is one higher than the highest job identifier
2780         value that any job can have on that system."
2781     REFERENCE
2782         "See Section 3.2 entitled 'The Job Tables'.
2783         See also jmGeneralNewestActiveJobIndex for the largest value of jmJobIndex.
2784         See JmJobSubmissionTypeTC for a limit on the size of this index if the agent represents it as
2785         an 8-digit decimal number."
2786     ::= { jmJobEntry 1 }
2787
2788 jmJobState OBJECT-TYPE
2789     SYNTAX      JmJobStateTC
2790     MAX-ACCESS  read-only
2791     STATUS      current
2792     DESCRIPTION
2793         "The current state of the job (pending, processing, completed, etc.). Agents SHALL
2794         implement only those states which are appropriate for the particular implementation. However,
2795         management applications SHALL be prepared to receive all the standard job states.
2796
2797         The final value for this object SHALL be one of: completed, canceled, or aborted. The
2798         minimum length of time that the agent SHALL maintain MIB data for a job in the completed,
2799         canceled, or aborted state before removing the job data from the jmJobIDTable and
2800         jmJobTable is specified by the value of the jmGeneralJobPersistence object."
2801     ::= { jmJobEntry 2 }
2802
2803 jmJobStateReasons1 OBJECT-TYPE
2804     SYNTAX      JmJobStateReasons1TC
2805     MAX-ACCESS  read-only
2806     STATUS      current
2807     DESCRIPTION
2808         "Additional information about the job's current state, i.e., information that augments the value of
2809         the job's jmJobState object.
2810

```

2811 Implementation of any reason values is OPTIONAL, but an agent SHOULD return any reason
 2812 information available. These values MAY be used with any job state or states for which the
 2813 reason makes sense. Furthermore, when implemented as with any MIB data, the agent SHALL
 2814 return these values when the reason applies and SHALL NOT return them when the reason no
 2815 longer applies whether the value of the job's **jmJobState** object changed or not. When the
 2816 agent cannot provide a reason for the current state of the job, the agent SHALL set the value of
 2817 the **jmJobStateReasons1** object and **jobStateReasonsN** attributes to **0**."

2818 REFERENCE
 2819 "The **jobStateReasonsN** ($N=2..4$) attributes provide further additional information about the
 2820 job's current state."
 2821 ::= { jmJobEntry 3 }

2822

2823 **jmNumberOfInterveningJobs** OBJECT-TYPE
 2824 SYNTAX **Integer32(-2..2147483647)**
 2825 MAX-ACCESS read-only
 2826 STATUS current
 2827 DESCRIPTION
 2828 "The number of jobs that are expected to be processed *before* this job is processed according to
 2829 the implementation's queuing algorithm if no other jobs were to be submitted. In other words,
 2830 this value is the job's queue position. The agent SHALL return a value of **0** for this attribute
 2831 while the job is processing."
 2832 ::= { jmJobEntry 4 }

2833

2834 **jmJobKOctetsRequested** OBJECT-TYPE
 2835 SYNTAX **Integer32(-2..2147483647)**
 2836 MAX-ACCESS read-only
 2837 STATUS current
 2838 DESCRIPTION
 2839 "The total size in K (1024) octets of the document(s) being requested to be processed in the job.
 2840 The agent SHALL round the actual number of octets up to the next highest K. Thus 0 octets
 2841 SHALL be represented as '0', 1-1024 octets SHALL be represented as '1', 1025-2048 SHALL
 2842 be represented as '2', etc.
 2843
 2844 In computing this value, the server/device SHALL *not* include the multiplicative factors
 2845 contributed by (1) the number of document copies, and (2) the number of job copies,
 2846 independent of whether the device can process multiple copies of the job or document without
 2847 making multiple passes over the job or document data and independent of whether the output is
 2848 collated or not. Thus the server/device computation is independent of the implementation."
 2849 ::= { jmJobEntry 5 }

2850

2851 **jmJobKOctetsProcessed** OBJECT-TYPE
 2852 SYNTAX **Integer32(-2..2147483647)**
 2853 MAX-ACCESS read-only
 2854 STATUS current
 2855 DESCRIPTION
 2856 "The current number of octets processed by the server or device measured in units of K (1024)
 2857 octets. The agent SHALL round the actual number of octets processed up to the next higher K.
 2858 Thus 0 octets SHALL be represented as '0', 1-1024 octets SHALL be represented as '1', 1025-

2859 2048 octets SHALL be '2', etc. For printing devices, this value is the number interpreted by the
 2860 page description language interpreter rather than what has been marked on media.
 2861

2862 For implementations where multiple copies are produced by the interpreter with only a single
 2863 pass over the data, the final value SHALL be equal to the value of the
 2864 **jmJobKOctetsRequested** object. For implementations where multiple copies are produced by
 2865 the interpreter by processing the data for each copy, the final value SHALL be a multiple of the
 2866 value of the **jmJobKOctetsRequested** object.
 2867

2868 NOTE - See the **impressionsCompletedCurrentCopy** and **pagesCompletedCurrentCopy**
 2869 attributes for attributes that are reset on each document copy.
 2870

2871 NOTE - The **jmJobKOctetsProcessed** object can be used with the **jmJobKOctetsRequested**
 2872 object to provide an indication of the relative progress of the job, provided that the
 2873 multiplicative factor is taken into account for some implementations of multiple copies."
 2874 ::= { jmJobEntry 6 }
 2875

2876 **jmJobImpressionsRequested** OBJECT-TYPE
 2877 SYNTAX **Integer32(-2..2147483647)**
 2878 MAX-ACCESS read-only
 2879 STATUS current
 2880 DESCRIPTION
 2881 "The number of impressions requested by this job to produce."
 2882 ::= { jmJobEntry 7 }
 2883

2884 **jmJobImpressionsCompleted** OBJECT-TYPE
 2885 SYNTAX **Integer32(-2..2147483647)**
 2886 MAX-ACCESS read-only
 2887 STATUS current
 2888 DESCRIPTION
 2889 "The current number of impressions completed for this job so far. For printing devices, the
 2890 impressions completed includes interpreting, marking, and stacking the output. For other types
 2891 of job services, the number of impressions completed includes the number of impressions
 2892 processed."
 2893 ::= { jmJobEntry 8 }
 2894

2895 **jmJobOwner** OBJECT-TYPE
 2896 SYNTAX **OCTET STRING(SIZE(0..63))**
 2897 MAX-ACCESS read-only
 2898 STATUS current
 2899 DESCRIPTION
 2900 "The coded character set name of the user that submitted the job. The method of assigning this
 2901 user name will be system and/or site specific but the method MUST insure that the name is
 2902 unique to the network that is visible to the client and target device.
 2903
 2904 This value SHOULD be the *authenticated* name of the user submitting the job."
 2905 REFERENCE
 2906 "See the OBJECT compliance macro for the minimum maximum length required for
 2907 conformance."

```

2908 ::= { jmJobEntry 9 }
2909
2910
2911
2912
2913 -- The Attribute Group (MANDATORY)
2914
2915 -- The jmAttributeGroup consists entirely of the jmAttributeTable.
2916 --
2917 -- Implementation of the two objects in this group is MANDATORY.
2918 -- See Section 3.1 entitled 'Conformance Considerations'.
2919 -- An agent SHALL implement any attribute if (1) the server or device
2920 -- supports the functionality represented by the attribute and (2) the
2921 -- information is available to the agent.
2922
2923 jmAttribute OBJECT IDENTIFIER ::= { jobmonMIBObjects 4 }
2924
2925 jmAttributeTable OBJECT-TYPE
2926     SYNTAX      SEQUENCE OF JmAttributeEntry
2927     MAX-ACCESS  not-accessible
2928     STATUS      current
2929     DESCRIPTION
2930         "The jmAttributeTable SHALL contain attributes of the job and document(s) for each job in a
2931         job set. Instead of allocating distinct objects for each attribute, each attribute is represented as a
2932         separate row in the jmAttributeTable."
2933     REFERENCE
2934         "The MANDATORY-GROUP macro specifies that this group is MANDATORY. An agent
2935         SHALL implement any attribute if (1) the server or device supports the functionality represented
2936         by the attribute and (2) the information is available to the agent. "
2937     ::= { jmAttribute 1 }
2938
2939 jmAttributeEntry OBJECT-TYPE
2940     SYNTAX      JmAttributeEntry
2941     MAX-ACCESS  not-accessible
2942     STATUS      current
2943     DESCRIPTION
2944         "Attributes representing information about the job and document(s) or resources required and/or
2945         consumed.
2946
2947         Each entry in the jmAttributeTable is a per-job entry with an extra index for each type of
2948         attribute (jmAttributeTypeIndex) that a job can have and an additional index
2949         (jmAttributeInstanceIndex) for those attributes that can have multiple instances per job. The
2950         jmAttributeTypeIndex object SHALL contain an enum type that indicates the type of attribute
2951         (see the JmAttributeTypeTC textual-convention). The value of the attribute SHALL be
2952         represented in either the jmAttributeValueAsInteger or jmAttributeValueAsOctets objects,
2953         and/or both, as specified in the JmAttributeTypeTC textual-convention.
2954
2955         The agent SHALL create rows in the jmAttributeTable as the server or device is able to
2956         discover the attributes either from the job submission protocol itself or from the document PDL.

```


2957 As the documents are interpreted, the interpreter MAY discover additional attributes and so the
 2958 agent adds additional rows to this table. As the attributes that represent resources are actually
 2959 consumed, the usage counter contained in the **jmAttributeValueAsInteger** object is
 2960 incremented according to the units indicated in the description of the **JmAttributeTypeTC**
 2961 enum.

2962

2963 The agent SHALL maintain each row in the **jmJobTable** for at least the minimum time after a
 2964 job completes as specified by the **jmGeneralAttributePersistence** object.

2965

2966 Zero or more entries SHALL exist in this table for each job in a job set."

2967 REFERENCE

2968 "See Section 3.3 entitled 'The Attribute Mechanism' for a description of the **jmAttributeTable**."

2969 INDEX { **jmGeneralJobSetIndex**, **jmJobIndex**, **jmAttributeTypeIndex**,
 2970 **jmAttributeInstanceIndex** }
 2971 ::= { jmAttributeTable 1 }
 2972

2973 **JmAttributeEntry** ::= SEQUENCE {
 2974 **jmAttributeTypeIndex** **JmAttributeTypeTC**,
 2975 **jmAttributeInstanceIndex** **Integer32(1..32767)**,
 2976 **jmAttributeValueAsInteger** **Integer32(-2..2147483647)**,
 2977 **jmAttributeValueAsOctets** **OCTET STRING(SIZE(0..63))**
 2978 }
 2979

2980 **jmAttributeTypeIndex** OBJECT-TYPE
 2981 SYNTAX **JmAttributeTypeTC**
 2982 MAX-ACCESS not-accessible
 2983 STATUS current
 2984 DESCRIPTION
 2985 "The type of attribute that this row entry represents.
 2986
 2987 The type MAY identify information about the job or document(s) or MAY identify a resource
 2988 required to process the job before the job start processing and/or consumed by the job as the job
 2989 is processed.
 2990
 2991 Examples of job and document attributes include: **jobCopiesRequested**,
 2992 **documentCopiesRequested**, **jobCopiesCompleted**, **documentCopiesCompleted**, **fileName**,
 2993 and **documentName**.
 2994
 2995 Examples of required and consumed resource attributes include: **pagesRequested**,
 2996 **pagesCompleted**, **mediumRequested**, and **mediumConsumed**, respectively."
 2997 ::= { jmAttributeEntry 1 }
 2998

2999 **jmAttributeInstanceIndex** OBJECT-TYPE
 3000 SYNTAX **Integer32(1..32767)**
 3001 MAX-ACCESS not-accessible
 3002 STATUS current
 3003 DESCRIPTION
 3004 "A running 16-bit index of the attributes of the same type for each job. For those attributes with
 3005 only a single instance per job, this index value SHALL be 1. For those attributes that are a

3006 single value per document, the index value SHALL be the document number, starting with **1** for
 3007 the first document in the job. Jobs with only a single document SHALL use the index value of
 3008 **1**. For those attributes that can have multiple values per job or per document, such as
 3009 **documentFormatIndex(37)** or **documentFormat(38)**, the index SHALL be a running index
 3010 for the job as a whole, starting at **1**."
 3011 ::= { jmAttributeEntry 2 }

3012 **jmAttributeValueAsInteger** OBJECT-TYPE
 3013 SYNTAX **Integer32(-2..2147483647)**
 3014 MAX-ACCESS read-only
 3015 STATUS current
 3016 DESCRIPTION
 3017 "The integer value of the attribute. The value of the attribute SHALL be represented as an
 3018 integer if the enum description in the **JmAttributeTypeTC** textual-convention definition has the
 3019 tag: 'INTEGER:'.
 3020
 3021 Depending on the enum definition, this object value MAY be an integer, a counter, an index, or
 3022 an enum, depending on the **jmAttributeTypeIndex** value. The units of this value are specified
 3023 in the enum description.
 3024
 3025 For those attributes that are accumulating job consumption as the job is processed as specified in
 3026 the **JmAttributeTypeTC** textual-convention, SHALL contain the final value after the job
 3027 completes processing, i.e., this value SHALL indicate the total usage of this resource made by
 3028 the job.
 3029
 3030 A monitoring application is able to copy this value to a suitable longer term storage for later
 3031 processing as part of an accounting system.
 3032
 3033 Since the agent MAY add attributes representing resources to this table while the job is waiting
 3034 to be processed or being processed, which can be a long time before any of the resources are
 3035 actually used, the agent SHALL set the value of the **jmAttributeValueAsInteger** object to **0**
 3036 for resources that the job has not yet consumed.
 3037
 3038 Attributes for which the concept of an integer value is meaningless, such as **fileName**,
 3039 **interpreter**, and **physicalDevice**, do *not* have the 'INTEGER:' tag in the **JmAttributeTypeTC**
 3040 definition and so an agent SHALL always return a value of '-1' to indicate 'other' for
 3041 **jmAttributeValueAsInteger**.
 3042
 3043 For attributes which do have the 'INTEGER:' tag in the **JmAttributeTypeTC** definition, if the
 3044 integer value is not (yet) known, the agent either SHALL not materialize the row in the
 3045 **jmAttributeTable** until the value is known or SHALL return a '-2' to represent an 'unknown'
 3046 counting integer value, a '0' to represent an 'unknown' index value, and a '2' to represent an
 3047 'unknown(2)' enum value."
 3048 ::= { jmAttributeEntry 3 }

3050 **jmAttributeValueAsOctets** OBJECT-TYPE
 3051 SYNTAX **OCTET STRING(SIZE(0..63))**
 3052 MAX-ACCESS read-only
 3053 STATUS current

3055 DESCRIPTION
3056 "The octet string value of the attribute. The value of the attribute SHALL be represented as an
3057 OCTET STRING if the enum description in the **JmAttributeTypeTC** textual-convention
3058 definition has the tag: 'OCTETS:'.
3059
3060 Depending on the enum definition, this object value MAY be a coded character set string (text)
3061 or a binary octet string, such as **DateAndTime**.
3062
3063 Attributes for which the concept of an octet string value is meaningless, such as
3064 **pagesCompleted**, do *not* have the tag 'OCTETS:' in the **JmAttributeTypeTC** definition and so
3065 the agent SHALL always return a zero length string for the value of the
3066 **jmAttributeValueAsOctets** object.
3067
3068 For attributes which do have the 'OCTETS:' tag in the **JmAttributeTypeTC** definition, if the
3069 OCTET STRING value is not (yet) known, the agent either SHALL not materialize the row in
3070 the **jmAttributeTable** until the value is known or SHALL return a zero-length string."
3071 ::= { jmAttributeEntry 4 }
3072

```

3073 -- Notifications and Trapping
3074 -- Reserved for the future
3075
3076 jobmonMIBNotifications OBJECT IDENTIFIER ::= { jobmonMIB 2 }
3077
3078
3079
3080 -- Conformance Information
3081
3082 jmMIBConformance OBJECT IDENTIFIER ::= { jobmonMIB 3 }
3083
3084 -- compliance statements
3085 jmMIBCompliance MODULE-COMPLIANCE
3086     STATUS current
3087     DESCRIPTION
3088         "The compliance statement for agents that implement the
3089         job monitoring MIB."
3090     MODULE -- this module
3091     MANDATORY-GROUPS {
3092         jmGeneralGroup, jmJobIDGroup, jmJobGroup, jmAttributeGroup }
3093
3094     OBJECT jmGeneralJobSetName
3095     SYNTAX OCTET STRING (SIZE(0..8))
3096     DESCRIPTION
3097         "Only 8 octets maximum string length NEED be supported by the agent."
3098
3099     OBJECT jmJobOwner
3100     SYNTAX OCTET STRING (SIZE(0..16))
3101     DESCRIPTION
3102         "Only 16 octets maximum string length NEED be supported by the agent."
3103
3104 -- There are no CONDITIONALLY MANDATORY or OPTIONAL groups.
3105
3106     ::= { jmMIBConformance 1 }
3107
3108 jmMIBGroups OBJECT IDENTIFIER ::= { jmMIBConformance 2 }
3109
3110 jmGeneralGroup OBJECT-GROUP
3111     OBJECTS {
3112         jmGeneralNumberOfActiveJobs, jmGeneralOldestActiveJobIndex,
3113         jmGeneralNewestActiveJobIndex, jmGeneralJobPersistence,
3114         jmGeneralAttributePersistence, jmGeneralJobSetName }
3115     STATUS current
3116     DESCRIPTION
3117         "The general group."
3118     ::= { jmMIBGroups 1 }
3119
3120 jmJobIDGroup OBJECT-GROUP
3121     OBJECTS {

```

```
3122     jmJobIDJobSetIndex, jmJobIDJobIndex }
3123     STATUS current
3124     DESCRIPTION
3125         "The job ID group."
3126     ::= { jmMIBGroups 2 }
3127
3128     jmJobGroup OBJECT-GROUP
3129     OBJECTS {
3130         jmJobState, jmJobStateReasons1, jmNumberOfInterveningJobs,
3131         jmJobKOctetsRequested, jmJobKOctetsProcessed, jmJobImpressionsRequested,
3132         jmJobImpressionsCompleted, jmJobOwner }
3133     STATUS current
3134     DESCRIPTION
3135         "The job group."
3136     ::= { jmMIBGroups 3 }
3137
3138     jmAttributeGroup OBJECT-GROUP
3139     OBJECTS {
3140         jmAttributeValueAsInteger, jmAttributeValueAsOctets }
3141     STATUS current
3142     DESCRIPTION
3143         "The attribute group."
3144     ::= { jmMIBGroups 4 }
3145
3146
3147     END
```

3148 **5. Appendix A - Implementing the Job Life Cycle**

3149 The job object has well-defined states and client operations that affect the transition between the
3150 job states. Internal server and device actions also affect the transitions of the job between the job
3151 states. These states and transitions are referred to as the job's *life cycle*.

3152 Not all implementations of job submission protocols have all of the states of the job model
3153 specified here. The job model specified here is intended to be a superset of most implementations.
3154 It is the purpose of the agent to map the particular implementation's job life cycle onto the one
3155 specified here. The agent MAY omit any states not implemented. Only the **processing** and
3156 **completed** states are required to be implemented by an agent. However, a conforming
3157 management application SHALL be prepared to accept any of the states in the job life cycle
3158 specified here, so that the management application can interoperate with any conforming agent.

3159 The job states are intended to be user visible. The agent SHALL make these states visible in the
3160 MIB, but only for the subset of job states that the implementation has. Some implementations
3161 MAY need to have sub-states of these user-visible states. The **jmJobStateReasons1** object and
3162 the **jobStateReasonsN** ($N=2..4$) attributes can be used to represent the sub-states of the jobs.

3163 Job states are intended to last a user-visible length of time in most implementations. However,
3164 some jobs may pass through some states in zero time in some situations and/or in some
3165 implementations.

3166 The job model does not specify how accounting and auditing is implemented, except to assume
3167 that accounting and auditing logs are separate from the job life cycle and last longer than job
3168 entries in the MIB. Jobs in the **completed**, **aborted**, or **canceled** states are not logs, since jobs in
3169 these states are accessible via SNMP protocol operations and SHALL be removed from the Job
3170 Monitoring MIB tables after a site-settable or implementation-defined period of time. An
3171 accounting application MAY copy accounting information incrementally to an accounting log as a
3172 job processes, or MAY be copied while the job is in the **canceled**, **aborted**, or **completed** states,
3173 depending on implementation. The same is true for auditing logs.

3174 **The jmJobState object specifies the standard job states. The normal job state transitions**
3175 **are shown in the state transition diagram presented in Table 1.**

3176 **6. APPENDIX B - Support of the Job Submission ID in Job Submission** 3177 **Protocols**

3178 This appendix lists the job submission protocols that support the concept of a job
3179 submission ID and indicates the attribute used in that job submission protocol.

3180 6.1 Hewlett-Packard's Printer Job Language (PJL)

3181 Hewlett-Packard's Printer Job Language provides job-level printer control and printer
3182 status information to applications. The PJL JOB command is used at the beginning of a
3183 print job and can include options applying only to that job. A PJL JOB command option
3184 has been defined to facilitate passing the **JobSubmissionID** with the print job, as required
3185 by the Job Monitoring MIB. The option is of the form:

```
3186         SUBMISSIONID = "id string"  
3187  
3188
```

3189 Where the "id string" is a string and SHALL be enclosed in double quotes. The format is
3190 as described for the **jmJobSubmissionID** object.

3191 The entire PJL JOB command with the optional parameter would be of the form:

```
3192         @PJL JOB SUBMISSIONID = "id string"  
3193  
3194
```

3195 See "Printer Job Language Technical Reference Manual", part number 5021-0328, from
3196 Hewlett-Packard for complete information on the PJL JOB command and the Printer Job
3197 Language.

3198 6.2 ISO DPA

3199 The ISO 10175 Document Printing Application (DPA) protocol specifies the "**job-client-**
3200 **id**" attribute that allows the client to supply a text string ID for each job.

3201 7. References

3202 [hr-mib] P. Grillo, S. Waldbusser, "Host Resources MIB", RFC 1514, September 1993

3203 [iana] J. Reynolds, and J. Postel, "Assigned Numbers", STD 2, RFC 1700, ISI, October
3204 1994.

3205 [iana-media-types] IANA Registration of MIME media types (MIME content
3206 types/subtypes). See <ftp://ftp.isi.edu/in-notes/iana/assignments/>

3207 [iso-dpa] ISO/IEC 10175 Document Printing Application (DPA). See
3208 <ftp://ftp.pwg.org/pub/pwg/dpa/>

3209 [ipp-model] Internet Printing Protocol (IPP), work in progress on the IETF standards
3210 track. See [draft-ietf-ipp-model-01.txt](#). See also <http://www.pwg.org/ipp/index.html>

3211 [mib-II] MIB-II, RFC 1213.

3212 [print-mib] The Printer MIB - RFC 1759, proposed IETF standard. Also an Internet-
3213 Draft on the standards track as a draft standard: [draft-ietf-printmib-mib-info-02.txt](#)

- 3214 [req-words] S. Bradner, "Keywords for use in RFCs to Indicate Requirement Levels",
3215 RFC 2119, March 1997.
- 3216 [rfc 2130] C. Weider, C. Preston, K. Simonsen, H. Alvestrand, R. Atkinson, M. Crispin,
3217 and P. Svanberg, "The Report of the IAB Character Set Workshop held 29 Feb-1 March,
3218 1997", April 1997, RFC 2130.
- 3219 [SMIv2-TC] J. Case, et al. "Textual Conventions for Version 2 of the Simple Network
3220 Management Protocol (SNMPv2)", RFC 1903, January 1996.
- 3221 [tipsi] IEEE 1284.1, Transport-independent Printer System Interface (TIPSI).
- 3222 [URI-spec] Berners-Lee, T., Masinter, L., McCahill, M. , "Uniform Resource Locators
3223 (URL)", RFC 1738, December, 1994.

3224 **8. Author's Addresses**

- 3225 Ron Bergman
3226 Dataproducts Corp.
3227 1757 Tapo Canyon Road
3228 Simi Valley, CA 93063-3394
3229
3230 Phone: 805-578-4421
3231 Fax: 805-578-4001
3232 Email: rbergman@dpc.com
3233
3234
- 3235 Tom Hastings
3236 Xerox Corporation, ESAE-231
3237 701 S. Aviation Blvd.
3238 El Segundo, CA 90245
3239
3240 Phone: 310-333-6413
3241 Fax: 310-333-5514
3242 EMail: hastings@cp10.es.xerox.com
3243
3244
- 3245 Scott A. Isaacson
3246 Novell, Inc.
3247 122 E 1700 S
3248 Provo, UT 84606
3249
3250 Phone: 801-861-7366
3251 Fax: 801-861-4025

3252 EMail: scott_isaacson@novell.com

3253

3254

3255 Harry Lewis

3256 IBM Corporation

3257 6300 Diagonal Hwy

3258 Boulder, CO 80301

3259

3260 Phone: (303) 924-5337

3261 Fax:

3262 Email: harryl@us.ibm.com

3263

3264

3265 Send comments to the printmib WG using the Job Monitoring Project (JMP)

3266 Mailing List: jmp@pwg.org

3267

3268 To learn how to subscribe, send email to: jmp-request@pwg.org

3269

3270 For further information, access the PWG web page under "JMP":

3271 <http://www.pwg.org/>

3272

3273 Other Participants:

3274 Chuck Adams - Tektronix

3275 Jeff Barnett - IBM

3276 Keith Carter, IBM Corporation

3277 Jeff Copeland - QMS

3278 Andy Davidson - Tektronix

3279 Roger deBry - IBM

3280 Mabry Dozier - QMS

3281 Lee Ferrel - Canon

3282 Steve Gebert - IBM

3283 Robert Herriot - Sun Microsystems Inc.

3284 Shige Kanemitsu - Kyocera

3285 David Kellerman - Northlake Software

3286 Rick Landau - Digital

3287 Harry Lewis - IBM

3288 Pete Loya - HP

3289 Ray Lutz - Cognisys

3290 Jay Martin - Underscore

3291 Mike MacKay, Novell, Inc.

3292 Stan McConnell - Xerox
3293 Carl-Uno Manros, Xerox, Corp.
3294 Pat Nogay - IBM
3295 Bob Pentecost - HP
3296 Rob Rhoads - Intel
3297 David Roach - Unisys
3298 Hiroyuki Sato - Canon
3299 Bob Setterbo - Adobe
3300 Gail Songer, EFI
3301 Mike Timperman - Lexmark
3302 Randy Turner - Sharp
3303 William Wagner - Digital Products
3304 Jim Walker - Dazel
3305 Chris Wellens - Interworking Labs
3306 Rob Whittle - Novell
3307 Don Wright - Lexmark
3308 Lloyd Young - Lexmark
3309 Atsushi Yuki - Kyocera
3310 Peter Zehler, Xerox, Corp.

3311 **9. INDEX**

3312 This index includes the textual conventions, the objects, and the attributes. Textual
 3313 conventions all start with the prefix: "JM" and end with the suffix: "TC". Objects all
 3314 starts with the prefix: "jm" followed by the group name. Attributes are identified with
 3315 enums, and so start with any lower case letter and have no special prefix.

		3349	jmGeneralNewestActiveJobIndex	64	
3316	—C—	3350	jmGeneralNumberOfActiveJobs	64	
		3351	jmGeneralOldestActiveJobIndex	64	
3317	colorantConsumed	51	3352	jmJobIDJobIndex	67
3318	colorantRequested	51	3353	jmJobIDJobSetIndex	67
		3354	jmJobImpressionsCompleted	71	
3319	—D—	3355	jmJobImpressionsRequested	71	
		3356	jmJobIndex	69	
3320	deviceNameRequested	44	3357	jmJobKOctetsProcessed	70
3321	documentCopiesCompleted	48	3358	jmJobKOctetsRequested	70
3322	documentCopiesRequested	47	3359	jmJobOwner	71
3323	documentFormat	45	3360	JmJobServiceTypesTC	54
3324	documentFormatIndex	45	3361	JmJobSourcePlatformTypeTC	30
3325	documentName	44	3362	jmJobState	69
		3363	jmJobStateReasons1	69	
3326	—F—	3364	JmJobStateReasons1TC	55	
		3365	JmJobStateReasons2TC	58	
3327	fileName	44	3366	JmJobStateReasons3TC	61
3328	finishing	46	3367	JmJobStateReasons4TC	62
3329	fullColorImpressionsCompleted	49	3368	JmJobStateTC	38
		3369	jmJobSubmissionID	67	
3330	—H—	3370	JmJobSubmissionTypeTC	36	
		3371	JmMediumTypeTC	35	
3331	highlightColorImpressionsCompleted	49	3372	jmNumberOfInterveningJobs	70
		3373	JmPrinterResolutionTC	33	
3332	—I—	3374	JmPrintQualityTC	32	
		3375	JmTimeStampTC	30	
3333	impressionsCompletedCurrentCopy	49	3376	JmTonerEconomyTC	34
3334	impressionsInterpreted	49	3377	jobAccountName	42
3335	impressionsSentToDevice	48	3378	jobComment	44
3336	impressionsSpooled	48	3379	jobCompletedTime	52
		3380	jobCopiesCompleted	47	
3337	—J—	3381	jobCopiesRequested	47	
		3382	jobHold	46	
3338	jmAttributeInstanceIndex	73	3383	jobHoldUntil	46
3339	jmAttributeTypeIndex	73	3384	jobKOctetsTransferred	48
3340	JmAttributeTypeTC	40	3385	jobName	42
3341	jmAttributeValueAsInteger	74	3386	jobOriginatingHost	43
3342	jmAttributeValueAsOctets	74	3387	jobPriority	45
3343	JmBooleanTC	34	3388	jobProcessAfterDateAndTime	45
3344	JmFinishingTC	31	3389	jobProcessingCPUtime	52
3345	jmGeneralAttributePersistence	65	3390	jobServiceTypes	43
3346	jmGeneralJobPersistence	65	3391	jobSourceChannelIndex	43
3347	jmGeneralJobSetIndex	63	3392	jobSourcePlatformType	43
3348	jmGeneralJobSetName	65	3393	jobStartedBeingHeldTime	52

3394	jobStartedProcessingTime	52	3413	printerResolutionRequested	47
3395	jobStateReasons2	41	3414	printerResolutionUsed	47
3396	jobStateReasons3	41	3415	printQualityRequested	46
3397	jobStateReasons4	41	3416	printQualityUsed	47
3398	jobSubmissionTime	51	3417	processingMessage	41
3399	jobSubmissionToServerTime	51			
			3418	—Q—	
3400	—M—		3419	queueNameRequested	44
3401	mediumConsumed	50			
3402	mediumRequested	50	3420	—S—	
			3421	serverAssignedJobName	42
3403	—N—		3422	sheetsCompleted	50
3404	numberOfDocuments	44	3423	sheetsCompletedCurrentCopy	50
			3424	sheetsRequested	50
3405	—O—		3425	sides	46
3406	other	41	3426	submittingApplicationName	43
3407	outputBin	46	3427	submittingServerName	43
			3428	—T—	
3408	—P—		3429	tonerDensityRequested	47
3409	pagesCompleted	49	3430	tonerDensityUsed	47
3410	pagesCompletedCurrentCopy	49	3431	tonerEcomonyRequested	47
3411	pagesRequested	49	3432	tonerEcomonyUsed	47
3412	physicalDevice	44			
3433					