

1 INTERNET-DRAFT

R. Bergman
Dataproducts Corp.
T. Hastings
Xerox Corporation
S. Isaacson
Novell, Inc.
H. Lewis
IBM Corp.
November 8, 1998

10 Job Monitoring MIB - V1.3
11 <draft-ietf-printmib-job-monitor-08.txt>

12
13 Status of this Memo

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

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

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

28 This Internet-Draft expires on May 8, 1998.

29
30

Abstract

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

45			
46		TABLE OF CONTENTS	
47	1	INTRODUCTION	7
48		1.1 Types of Information in the MIB	7
49		1.2 Types of Job Monitoring Applications	9
50	2	TERMINOLOGY AND JOB MODEL	10
51		2.1 System Configurations for the Job Monitoring MIB	13
52		2.1.1 Configuration 1 - client-printer	13
53		2.1.2 Configuration 2 - client-server-printer - agent in the	
54		server	14
55		2.1.3 Configuration 3 - client-server-printer - client monitors	
56		printer agent and server	15
57	3	MANAGED OBJECT USAGE	17
58		3.1 Conformance Considerations	17
59		3.1.1 Conformance Terminology	17
60		3.1.2 Agent Conformance Requirements	17
61		3.1.2.1 MIB II System Group objects	18
62		3.1.2.2 MIB II Interface Group objects	18
63		3.1.2.3 Printer MIB objects	18
64		3.1.3 Job Monitoring Application Conformance Requirements	18
65		3.2 The Job Tables and the Oldest Active and Newest Active Indexes	19
66		3.3 The Attribute Mechanism and the Attribute Table(s)	21
67		3.3.1 Conformance of Attribute Implementation	22
68		3.3.2 Useful, 'Unknown', and 'Other' Values for Objects and	
69		Attributes	22
70		3.3.3 Index Value Attributes	23
71		3.3.4 Data Sub-types and Attribute Naming Conventions	23
72		3.3.5 Single-Value (Row) Versus Multi-Value (MULTI-ROW)	
73		Attributes	24
74		3.3.6 Requested Objects and Attributes	24
75		3.3.7 Consumption Attributes	24
76		3.3.8 Attribute Specifications	25
77		3.3.9 Job State Reason bit definitions	44
78		3.3.9.1 JmJobStateReasons1TC specification	45
79		3.3.9.2 JmJobStateReasons2TC specification	49
80		3.3.9.3 JmJobStateReasons3TC specification	53
81		3.3.9.4 JmJobStateReasons4TC specification	53
82		3.4 Monitoring Job Progress	53
83		3.5 Job Identification	58

84	3.5.1	The Job Submission ID specifications	59
85	3.6	Internationalization Considerations	63
86	3.6.1	Text generated by the server or device	63
87	3.6.2	Text supplied by the job submitter	64
88	3.6.3	'DateAndTime' for representing the date and time	65
89	3.7	IANA and PWG Registration Considerations	65
90	3.7.1	PWG Registration of enums	66
91	3.7.1.1	Type 1 enumerations	66
92	3.7.1.2	Type 2 enumerations	66
93	3.7.1.3	Type 3 enumeration	67
94	3.7.2	PWG Registration of type 2 bit values	67
95	3.7.3	PWG Registration of Job Submission Id Formats	67
96	3.7.4	PWG Registration of MIME types/sub-types for document-	
97		formats	68
98	3.8	Security Considerations	68
99	3.8.1	Read-Write objects	68
100	3.8.2	Read-Only Objects In Other User's Jobs	68
101	3.9	Notifications	68
102	4	MIB SPECIFICATION	69
103		Textual conventions for this MIB module	71
104		JmUTF8StringTC	71
105		JmJobStringTC	71
106		JmNaturalLanguageTagTC	71
107		JmTimeStampTC	71
108		JmJobSourcePlatformTypeTC	72
109		JmFinishingTC	73
110		JmPrintQualityTC	74
111		JmPrinterResolutionTC	74
112		JmTonerEconomyTC	75
113		JmBooleanTC	75
114		JmMediumTypeTC	75
115		JmJobCollationTypeTC	76
116		JmJobSubmissionIDTypeTC	77
117		JmJobStateTC	78
118		JmAttributeTypeTC	81
119		JmJobServiceTypesTC	84
120		JmJobStateReasons1TC	86
121		JmJobStateReasons2TC	86
122		JmJobStateReasons3TC	86
123		JmJobStateReasons4TC	87
124		The General Group (MANDATORY)	88
125		jmGeneralJobSetIndex (Int32(1..32767))	89
126		jmGeneralNumberOfActiveJobs (Int32(0..))	89
127		jmGeneralOldestActiveJobIndex (Int32(0..))	90

128	jmGeneralNewestActiveJobIndex	(Int32(0..))	90
129	jmGeneralJobPersistence	(Int32(15..))	91
130	jmGeneralAttributePersistence	(Int32(15..))	91
131	jmGeneralJobSetName	(UTF8String63)	92
132	The Job ID Group (MANDATORY)		92
133	jmJobSubmissionID	(OCTET STRING(SIZE(48)))	94
134	jmJobIDJobSetIndex	(Int32(0..32767))	95
135	jmJobIDJobIndex	(Int32(0..))	95
136	The Job Group (MANDATORY)		95
137	jmJobIndex	(Int32(1..))	97
138	jmJobState	(JmJobStateTC)	97
139	jmJobStateReasons1	(JmJobStateReasons1TC)	98
140	jmNumberOfInterveningJobs	(Int32(-2..))	98
141	jmJobKOctetsPerCopyRequested	(Int32(-2..))	99
142	jmJobKOctetsProcessed	(Int32(-2..))	99
143	jmJobImpressionsPerCopyRequested	(Int32(-2..))	100
144	jmJobImpressionsCompleted	(Int32(-2..))	100
145	jmJobOwner	(JobString63)	101
146	The Attribute Group (MANDATORY)		101
147	jmAttributeTypeIndex	(JmAttributeTypeTC)	104
148	jmAttributeInstanceIndex	(Int32(1..32767))	104
149	jmAttributeValueAsInteger	(Int32(-2..))	105
150	jmAttributeValueAsOctets	(Octets63)	106
151	The Mirror Attribute Group (OPTIONAL)		107
152	jmMirrorAttrTypeIndex	(JmAttributeTypeTC)	108
153	jmMirrorAttrInstanceIndex	(Int32(1..32767))	108
154	jmMirrorAttrValueAsInteger	(Int32(-2..))	108
155	jmMirrorAttrValueAsOctets	(Octets63)	108
156	5	APPENDIX A - IMPLEMENTING THE JOB LIFE CYCLE	113
157	6	APPENDIX B - SUPPORT OF JOB SUBMISSION PROTOCOLS	114
158	7	REFERENCES	114
159	8	AUTHOR'S ADDRESSES	116
160	9	CHANGE HISTORY	118
161	9.1	Changes to produce version 1.3, dated November 8, 1998	118
162	9.2	Changes to produce version 1.2, dated October 2, 1998	118
163	9.3	Changes to produce version 1.1, dated October 1, 1998	119

164 10 INDEX
165

120

166 Job Monitoring MIB

167 1 Introduction

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

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

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

204 **1.1 Types of Information in the MIB**

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

208 User:

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

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

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

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

219 Operator:

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

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

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

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

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

232 Capacity Planner:

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

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

237 Accountant:

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

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

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

249 1.2 Types of Job Monitoring Applications

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

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

282 The MIB provides a set of objects that represent a compatible subset of
283 job and document attributes of the ISO DPA standard[iso-dpa] and the
284 Internet Printing Protocol (IPP)[ipp-model], so that coherence is
285 maintained between these two protocols and the information presented to
286 end users and system operators by monitoring applications. However,
287 the job monitoring MIB is intended to be used with printers that
288 implement other job submitting and management protocols, such as IEEE
289 1284.1 (TIPSI)[tipsi], as well as with ones that do implement ISO DPA.

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

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

295 2 Terminology and Job Model

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

394 Monitor: Short for Job Monitoring Application.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

431 **2.1 System Configurations for the Job Monitoring MIB**

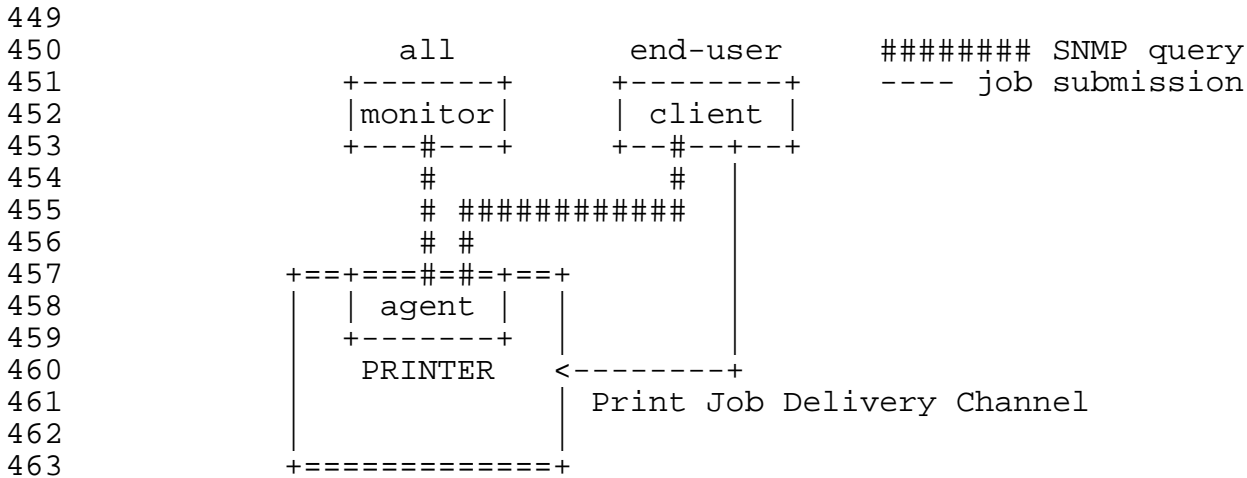
432 This section enumerates the three configurations in which the Job
433 Monitoring MIB is intended to be used. To simplify the pictures, the
434 *devices* are shown as *printers*. See section 1.1 entitled "Types of
435 Information in the MIB".

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

439 2.1.1 Configuration 1 - client-printer

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

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



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

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

- 467 1. Multiple clients MAY submit jobs to a printer.
- 468 2. Multiple clients MAY monitor a printer.
- 469 3. Multiple monitors MAY monitor a printer.
- 470 4. A client MAY submit jobs to multiple printers.
- 471 5. A monitor MAY monitor multiple printers.

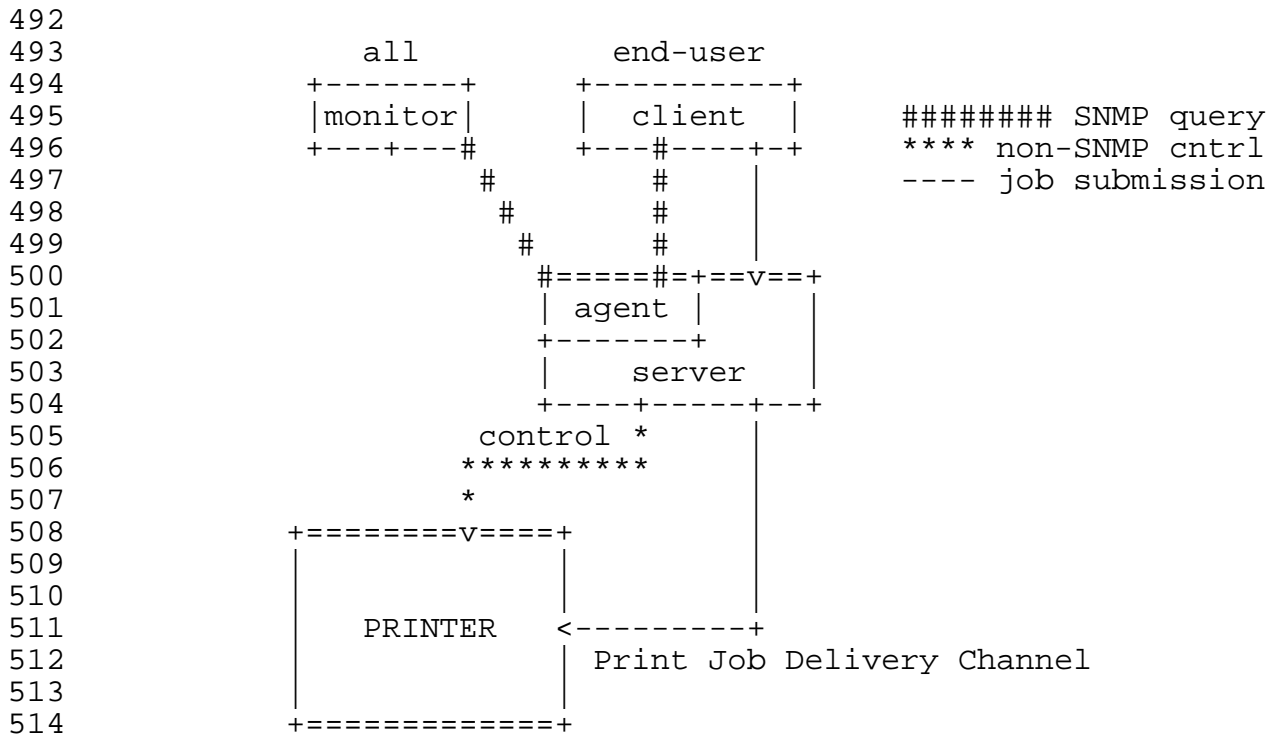
472 2.1.2 Configuration 2 - client-server-printer - agent in the server

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

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

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

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



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

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

- 519 1. Multiple clients MAY submit jobs to a server.
- 520 2. Multiple clients MAY monitor a server.
- 521 3. Multiple monitors MAY monitor a server.
- 522 4. A client MAY submit jobs to multiple servers.
- 523 5. A monitor MAY monitor multiple servers.
- 524 6. Multiple servers MAY submit jobs to a printer.
- 525 7. Multiple servers MAY control a printer.

526 2.1.3 Configuration 3 - client-server-printer - client monitors printer
527 agent and server

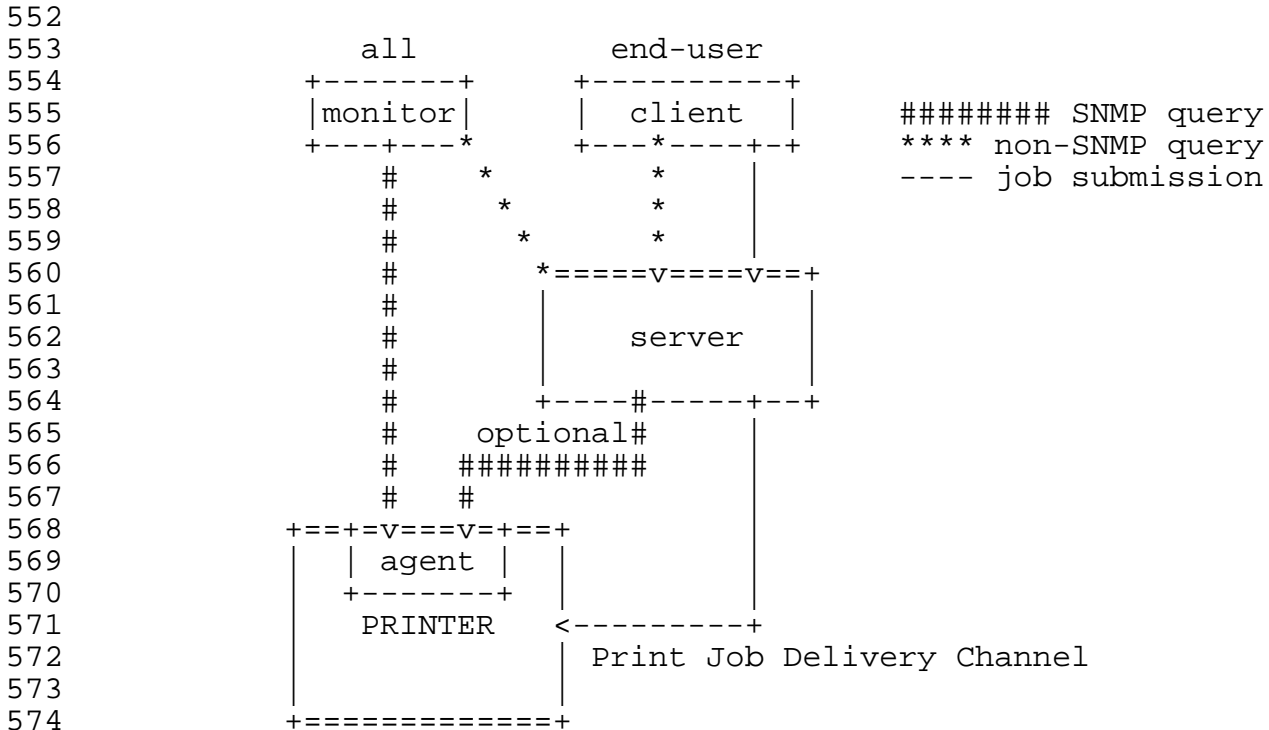
528 In the client-server-printer configuration 3, the client(s) submit jobs
529 to an intermediate server by some network connection, *not* directly to
530 the printer. That server does *not* contain a Job Monitoring MIB agent.

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

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

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

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



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

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

- 579 1. Multiple clients MAY submit jobs to a server.
- 580 2. Multiple clients MAY monitor a server.
- 581 3. Multiple monitors MAY monitor a server.
- 582 4. A client MAY submit jobs to multiple servers.
- 583 5. A monitor MAY monitor multiple servers.
- 584 6. Multiple servers MAY submit jobs to a printer.
- 585 7. Multiple servers MAY control a printer.

586 3 Managed Object Usage

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

588 **3.1 Conformance Considerations**

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

592 3.1.1 Conformance Terminology

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

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

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

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

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

608 3.1.2 Agent Conformance Requirements

609 A conforming agent:

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

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

622 3.1.2.1 MIB II System Group objects

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

626 3.1.2.2 MIB II Interface Group objects

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

630 3.1.2.3 Printer MIB objects

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

638 3.1.3 Job Monitoring Application Conformance Requirements

639 A conforming job monitoring application:

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

658 **3.2 The Job Tables and the Oldest Active and Newest Active Indexes**

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

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

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

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

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

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

682 It is recommended that agents that are providing access to
683 servers/devices that already allocate job-identifiers for jobs as
684 integers use the same integer value for the jmJobIndex. Then
685 management applications using this MIB and applications using other
686 protocols will see the same job identifiers for the same jobs. Agents
687 providing access to systems that contain jobs with a job identifier of
688 0 SHALL map the job identifier value 0 to a jmJobIndex value that is
689 one higher than the highest job identifier value that any job can have
690 on that system. Then only job 0 will have a different job-identifier
691 value than the job's jmJobIndex value.

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

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

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

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

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

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

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

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

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

741 3.3 The Attribute Mechanism and the Attribute Table(s)

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

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

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

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

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

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

766 With this order of table indexing, an application can obtain selected
767 attributes of a number of jobs using SNMPv1 GetNext or SNMPv2 GetBulk.

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

776 NOTE - The table of contents lists all the attributes in order. This
777 order is the order of enum assignments which is the order that the SNMP
778 GetNext operation returns attributes. Most attributes apply to all
779 three configurations covered by this MIB specification (see section 2.1
780 entitled "System Configurations for the Job Monitoring MIB"). Those

781 attributes that apply to a particular configuration are indicated as
782 'Configuration n:' and SHALL NOT be used with other configurations.

783 3.3.1 Conformance of Attribute Implementation

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

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

793 3.3.2 Useful, 'Unknown', and 'Other' Values for Objects and Attributes

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

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

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

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

822 attributes, the agent SHALL always return a '-1' value for the
823 jmAttributeValueAsInteger object.

824 3.3.3 Index Value Attributes

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

833 3.3.4 Data Sub-types and Attribute Naming Conventions

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

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

853

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

854

855 3.3.5 Single-Value (Row) Versus Multi-Value (MULTI-ROW) Attributes

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

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

871 3.3.6 Requested Objects and Attributes

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

882 3.3.7 Consumption Attributes

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

891 3.3.8 Attribute Specifications

892 This section specifies the job attributes.

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

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

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

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

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

914 NOTE: No attribute name exceeds 31 characters.

915 The standard attribute types are:

916	jmAttributeTypeIndex	Datatype
917	-----	-----
918		
919		
920	other(1),	Integer32 (-2..2147483647)
921		AND/OR
922		OCTET STRING(SIZE(0..63))
923	INTEGER: and/or OCTETS:	An attribute that is not in the
924	list and/or that has not been	approved and registered with
925	the PWG.	

926 ++++++
927 + Job State attributes
928 +
929 + The following attributes specify the state of a job.
930 ++++++

931
932 jobStateReasons2(3), JmJobStateReasons2TC
933 INTEGER: Additional information about the job's current
934 state that augments the jmJobState object. See the
935 description under the JmJobStateReasons1TC textual-
936 convention.
937

938 jobStateReasons3(4), JmJobStateReasons3TC
939 INTEGER: Additional information about the job's current
940 state that augments the jmJobState object. See the
941 description under JmJobStateReasons1TC textual-convention.
942

943 jobStateReasons4(5), JmJobStateReasons4TC
944 INTEGER: Additional information about the job's current
945 state that augments the jmJobState object. See the
946 description under JmJobStateReasons1TC textual-convention.
947

948 processingMessage(6), JmUTF8StringTC (SIZE(0..63))
949 OCTETS: MULTI-ROW: A coded character set message that is
950 generated by the server or device during the processing of
951 the job as a simple form of processing log to show progress
952 and any problems. The natural language of each value is
953 specified by the corresponding
954 processingMessageNaturalLangTag(7) value.
955

956 NOTE - This attribute is intended for such conditions as
957 interpreter messages, rather than being the printable form
958 of the jmJobState and jmJobStateReasons1 objects and
959 jobStateReasons2, jobStateReasons3, and jobStateReasons4
960 attributes. In order to produce a localized printable form
961 of these job state objects/attribute, a management
962 application SHOULD produce a message from their enum and
963 bit values.
964

965 NOTE - There is no job description attribute in IPP/1.0
966 that corresponds to this attribute and this attribute does
967 not correspond to the IPP/1.0 'job-state-message' job
968 description attribute, which is just a printable form of
969 the IPP 'job-state' and 'job-state-reasons' job attributes.
970

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

973
974 processingMessageNaturalLangTag(7), OCTET STRING(SIZE(0..63))
975 OCTETS: MULTI-ROW: The natural language of the
976 corresponding processingMessage(6) attribute value. See
977 section 3.6.1, entitled 'Text generated by the server or
978 device'.
979

980 If the agent does not know the natural language of the job
981 processing message, the agent SHALL either (1) return a
982 zero length string value for the
983 processingMessageNaturalLangTag(7) attribute or (2) not
984 return the processingMessageNaturalLangTag(7) attribute for
985 the job.
986

987 There is no restriction for the same tag occurring in
988 multiple rows, since when this attribute is implemented, it
989 SHOULD have a value row for each corresponding
990 processingMessage(6) attribute value row.
991

992 jobCodedCharSet(8), CodedCharSet
993 INTEGER: The MIBenum identifier of the coded character set
994 that the agent is using to represent coded character set
995 objects and attributes of type 'JmJobStringTC'. These
996 coded character set objects and attributes are either: (1)
997 supplied by the job submitting client or (2) defaulted by
998 the server or device when omitted by the job submitting
999 client. The agent SHALL represent these objects and
1000 attributes in the MIB either (1) in the coded character set
1001 as they were submitted or (2) MAY convert the coded
1002 character set to another coded character set or encoding
1003 scheme as identified by the jobCodedCharSet(8) attribute.
1004 See section 3.6.2, entitled 'Text supplied by the job
1005 submitter'.
1006

1007 These MIBenum values are assigned by IANA [IANA-charsets]
1008 when the coded character sets are registered. The coded
1009 character set SHALL be one of the ones registered with IANA
1010 [IANA] and the enum value uses the CodedCharSet textual-
1011 convention from the Printer MIB. See the JmJobStringTC
1012 textual-convention.
1013

1014 If the agent does not know what coded character set was
1015 used by the job submitting client, the agent SHALL either
1016 (1) return the 'unknown(2)' value for the
1017 jobCodedCharSet(8) attribute or (2) not return the
1018 jobCodedCharSet(8) attribute for the job.

1019 jobNaturalLanguageTag(9), OCTET STRING(SIZE(0..63))
1020 OCTETS: The natural language of the job attributes supplied
1021 by the job submitter or defaulted by the server or device
1022 for the job, i.e., all objects and attributes represented
1023 by the 'JmJobStringTC' textual-convention, such as jobName,
1024 mediumRequested, etc. See Section 3.6.2, entitled 'Text
1025 supplied by the job submitter'.
1026
1027 If the agent does not know what natural language was used
1028 by the job submitting client, the agent SHALL either (1)
1029 return a zero length string value for the
1030 jobNaturalLanguageTag(9) attribute or (2) not return
1031 jobNaturalLanguageTag(9) attribute for the job.
1032
1033 +++++
1034 + Job Identification attributes
1035 +
1036 + The following attributes help an end user, a system
1037 + operator, or an accounting program identify a job.
1038 +++++
1039
1040 jobURI(20), OCTET STRING(SIZE(0..63))
1041 OCTETS: MULTI-ROW: The job's Universal Resource
1042 Identifier (URI) [RFC-1738]. See IPP [ipp-model] for
1043 example usage.
1044
1045 NOTE - The agent may be able to generate this value on each
1046 SNMP Get operation from smaller values, rather than having
1047 to store the entire URI.
1048
1049 If the URI exceeds 63 octets, the agent SHALL use multiple
1050 values, with the next 63 octets coming in the second value,
1051 etc.
1052
1053 NOTE - IPP [ipp-model] has a 1023-octet maximum length for
1054 a URI, though the URI standard itself and HTTP/1.1 specify
1055 no maximum length.
1056
1057 jobAccountName(21), OCTET STRING(SIZE(0..63))
1058 OCTETS: Arbitrary binary information which MAY be coded
1059 character set data or encrypted data supplied by the
1060 submitting user for use by accounting services to allocate
1061 or categorize charges for services provided, such as a
1062 customer account name or number.
1063
1064 NOTE: This attribute NEED NOT be printable characters.
1065

1066 serverAssignedJobName(22), JmJobStringTC (SIZE(0..63))
1067 OCTETS: Configuration 3 only: The human readable string
1068 name, number, or ID of the job as assigned by the server
1069 that submitted the job to the device that the agent is
1070 providing access to with this MIB.
1071
1072 NOTE - This attribute is intended for enabling a user to
1073 find his/her job that a server submitted to a device when
1074 either the client does not support the jmJobSubmissionID or
1075 the server does not pass the jmJobSubmissionID through to
1076 the device.
1077
1078 jobName(23), JmJobStringTC (SIZE(0..63))
1079 OCTETS: The human readable string name of the job as
1080 assigned by the submitting user to help the user
1081 distinguish between his/her various jobs. This name does
1082 not need to be unique.
1083
1084 This attribute is intended for enabling a user or the
1085 user's application to convey a job name that MAY be printed
1086 on a start sheet, returned in a query result, or used in
1087 notification or logging messages.
1088
1089 In order to assist users to find their jobs for job
1090 submission protocols that don't supply a jmJobSubmissionID,
1091 the agent SHOULD maintain the jobName attribute for the
1092 time specified by the jmGeneralJobPersistence object,
1093 rather than the (shorter) jmGeneralAttributePersistence
1094 object.
1095
1096 If this attribute is not specified when the job is
1097 submitted, no job name is assumed, but implementation
1098 specific defaults are allowed, such as the value of the
1099 documentName attribute of the first document in the job or
1100 the fileName attribute of the first document in the job.
1101
1102 The jobName attribute is distinguished from the jobComment
1103 attribute, in that the jobName attribute is intended to
1104 permit the submitting user to distinguish between different
1105 jobs that he/she has submitted. The jobComment attribute
1106 is intended to be free form additional information that a
1107 user might wish to use to communicate with himself/herself,
1108 such as a reminder of what to do with the results or to
1109 indicate a different set of input parameters were tried in
1110 several different job submissions.
1111

1112 jobServiceTypes(24), JmJobServiceTypesTC
1113 INTEGER: Specifies the type(s) of service to which the job
1114 has been submitted (print, fax, scan, etc.). The service
1115 type is bit encoded with each job service type so that more
1116 general and arbitrary services can be created, such as
1117 services with more than one destination type, or ones with
1118 only a source or only a destination. For example, a job
1119 service might scan, faxOut, and print a single job. In
1120 this case, three bits would be set in the jobServiceTypes
1121 attribute, corresponding to the hexadecimal values: 0x8 +
1122 0x20 + 0x4, respectively, yielding: 0x2C.
1123
1124 Whether this attribute is set from a job attribute supplied
1125 by the job submission client or is set by the recipient job
1126 submission server or device depends on the job submission
1127 protocol. This attribute SHALL be implemented if the
1128 server or device has other types in addition to or instead
1129 of printing.
1130
1131 One of the purposes of this attribute is to permit a
1132 requester to filter out jobs that are not of interest. For
1133 example, a printer operator may only be interested in jobs
1134 that include printing.
1135
1136 jobSourceChannelIndex(25), Integer32 (0..2147483647)
1137 INTEGER: The index of the row in the associated Printer
1138 MIB[print-mib] of the channel which is the source of the
1139 print job.
1140
1141 jobSourcePlatformType(26), JmJobSourcePlatformTypeTC
1142 INTEGER: The source platform type of the immediate
1143 upstream submitter that submitted the job to the server
1144 (configuration 2) or device (configuration 1 and 3) to
1145 which the agent is providing access. For configuration 1,
1146 this is the type of the client that submitted the job to
1147 the device; for configuration 2, this is the type of the
1148 client that submitted the job to the server; and for
1149 configuration 3, this is the type of the server that
1150 submitted the job to the device.
1151
1152 submittingServerName(27), JmJobStringTC (SIZE(0..63))
1153 OCTETS: For configuration 3 only: The administrative name
1154 of the server that submitted the job to the device.
1155
1156 submittingApplicationName(28), JmJobStringTC (SIZE(0..63))
1157 OCTETS: The name of the client application (not the server
1158 in configuration 3) that submitted the job to the server or
1159 device.
1160

1161 jobOriginatingHost(29), JmJobStringTC (SIZE(0..63))
1162 OCTETS: The name of the client host (not the server host
1163 name in configuration 3) that submitted the job to the
1164 server or device.
1165

1166 deviceNameRequested(30), JmJobStringTC (SIZE(0..63))
1167 OCTETS: The administratively defined coded character set
1168 name of the target device requested by the submitting user.
1169 For configuration 1, its value corresponds to the Printer
1170 MIB[print-mib]: prtGeneralPrinterName object. For
1171 configuration 2 and 3, its value is the name of the logical
1172 or physical device that the user supplied to indicate to
1173 the server on which device(s) they wanted the job to be
1174 processed.
1175

1176 queueNameRequested(31), JmJobStringTC (SIZE(0..63))
1177 OCTETS: The administratively defined coded character set
1178 name of the target queue requested by the submitting user.
1179 For configuration 1, its value corresponds to the queue in
1180 the device for which the agent is providing access. For
1181 configuration 2 and 3, its value is the name of the queue
1182 that the user supplied to indicate to the server on which
1183 device(s) they wanted the job to be processed.
1184

1185 NOTE - typically an implementation SHOULD support either
1186 the deviceNameRequested or queueNameRequested attribute,
1187 but not both.
1188

1189 physicalDevice(32), hrDeviceIndex
1190 AND/OR
1191 JmUTF8StringTC (SIZE(0..63))
1192 INTEGER: MULTI-ROW: The index of the physical device MIB
1193 instance requested/used, such as the Printer MIB[print-
1194 mib]. This value is an hrDeviceIndex value. See the Host
1195 Resources MIB[hr-mib].
1196

1197 AND/OR
1198

1199 OCTETS: MULTI-ROW: The name of the physical device to
1200 which the job is assigned.
1201

1202 numberOfDocuments(33), Integer32 (-2..2147483647)
1203 INTEGER: The number of documents in this job.
1204

1205 The agent SHOULD return this attribute if the job has more
1206 than one document.
1207


```

1249     documentFormat(38),                               PrtInterpreterLangFamilyTC
1250                                                     AND/OR
1251                                                     OCTET STRING(SIZE(0..63))
1252     INTEGER: MULTI-ROW: The interpreter language family
1253     corresponding to the Printer MIB[print-mib]
1254     prtInterpreterLangFamily object, that this job
1255     requires/uses. A document or a job MAY use more than one
1256     PDL or control language.
1257
1258     AND/OR
1259
1260     OCTETS: MULTI-ROW: The document format registered as a
1261     media type[iana-media-types], i.e., the name of the MIME
1262     content-type/subtype. Examples: 'application/postscript',
1263     'application/vnd.hp-PCL', 'application/pdf', 'text/plain'
1264     (US-ASCII SHALL be assumed), 'text/plain; charset=iso-8859-
1265     1', and 'application/octet-stream'. The IPP 'document-
1266     format' job attribute uses these same values with the same
1267     semantics. See the IPP [ipp-model] 'mimeMediaType'
1268     attribute syntax and the document-format attribute for
1269     further examples and explanation.
1270
1271     ++++++
1272     + Job Parameter attributes
1273     +
1274     + The following attributes represent input parameters
1275     + supplied by the submitting client in the job submission
1276     + protocol.
1277     ++++++
1278
1279     jobPriority(50),                                     Integer32 (-2..100)
1280     INTEGER: The priority for scheduling the job. It is used
1281     by servers and devices that employ a priority-based
1282     scheduling algorithm.
1283
1284     A higher value specifies a higher priority. The value 1 is
1285     defined to indicate the lowest possible priority (a job
1286     which a priority-based scheduling algorithm SHALL pass over
1287     in favor of higher priority jobs). The value 100 is
1288     defined to indicate the highest possible priority.
1289     Priority is expected to be evenly or 'normally' distributed
1290     across this range. The mapping of vendor-defined priority
1291     over this range is implementation-specific. -2 indicates
1292     unknown.
1293

```

1294 jobProcessAfterDateAndTime(51), DateAndTime (SNMPv2-TC)
1295 OCTETS: The calendar date and time of day after which the
1296 job SHALL become a candidate to be scheduled for
1297 processing. If the value of this attribute is in the
1298 future, the server SHALL set the value of the job's
1299 jmJobState object to pendingHeld and add the
1300 jobProcessAfterSpecified bit value to the job's
1301 jmJobStateReasons1 object. When the specified date and
1302 time arrives, the server SHALL remove the
1303 jobProcessAfterSpecified bit value from the job's
1304 jmJobStateReasons1 object and, if no other reasons remain,
1305 SHALL change the job's jmJobState object to pending.
1306
1307 jobHold(52), JmBooleanTC
1308 INTEGER: If the value is 'true(4)', a client has
1309 explicitly specified that the job is to be held until
1310 explicitly released. Until the job is explicitly released
1311 by a client, the job SHALL be in the pendingHeld state with
1312 the jobHoldSpecified value in the jmJobStateReasons1
1313 attribute.
1314
1315 jobHoldUntil(53), JmJobStringTC (SIZE(0..63))
1316 OCTETS: The named time period during which the job SHALL
1317 become a candidate for processing, such as 'evening',
1318 'night', 'weekend', 'second-shift', 'third-shift', etc.,
1319 (supported values configured by the system administrator).
1320 See IPP [ipp-model] for the standard keyword values. Until
1321 that time period arrives, the job SHALL be in the
1322 pendingHeld state with the jobHoldUntilSpecified value in
1323 the jmJobStateReasons1 object. The value 'no-hold' SHALL
1324 indicate explicitly that no time period has been specified;
1325 the absence of this attribute SHALL indicate implicitly
1326 that no time period has been specified.
1327
1328 outputBin(54), Integer32 (0..2147483647)
1329 AND/OR
1330 JmJobStringTC (SIZE(0..63))
1331 INTEGER: MULTI-ROW: The output subunit index in the
1332 Printer MIB[print-mib]
1333
1334 AND/OR
1335
1336 OCTETS: MULTI-ROW: the name or number (represented as
1337 ASCII digits) of the output bin to which all or part of the
1338 job is placed in.
1339

```
1340 sides(55), Integer32 (-2..2)
1341     INTEGER: MULTI-ROW: The number of sides, '1' or '2', that
1342     any document in this job requires/used.
1343
1344 finishing(56), JmFinishingTC
1345     INTEGER: MULTI-ROW: Type of finishing that any document
1346     in this job requires/used.
1347
1348
1349 ++++++
1350 + Image Quality attributes (requested and consumed)
1351 +
1352 + For devices that can vary the image quality.
1353 ++++++
1354
1355 printQualityRequested(70), JmPrintQualityTC
1356     INTEGER: MULTI-ROW: The print quality selection requested
1357     for a document in the job for printers that allow quality
1358     differentiation.
1359
1360 printQualityUsed(71), JmPrintQualityTC
1361     INTEGER: MULTI-ROW: The print quality selection actually
1362     used by a document in the job for printers that allow
1363     quality differentiation.
1364
1365 printerResolutionRequested(72), JmPrinterResolutionTC
1366     OCTETS: MULTI-ROW: The printer resolution requested for a
1367     document in the job for printers that support resolution
1368     selection.
1369
1370 printerResolutionUsed(73), JmPrinterResolutionTC
1371     OCTETS: MULTI-ROW: The printer resolution actually used
1372     by a document in the job for printers that support
1373     resolution selection.
1374
1375 tonerEcomonyRequested(74), JmTonerEcomonyTC
1376     INTEGER: MULTI-ROW: The toner economy selection requested
1377     for documents in the job for printers that allow toner
1378     economy differentiation.
1379
1380 tonerEcomonyUsed(75), JmTonerEcomonyTC
1381     INTEGER: MULTI-ROW: The toner economy selection actually
1382     used by documents in the job for printers that allow toner
1383     economy differentiation.
1384
1385 tonerDensityRequested(76) Integer32 (-2..100)
1386     INTEGER: MULTI-ROW: The toner density requested for a
1387     document in this job for devices that can vary toner
1388     density levels. Level 1 is the lowest density and level
1389     100 is the highest density level. Devices with a smaller
1390     range, SHALL map the 1-100 range evenly onto the
1391     implemented range.
```

1392
1393 tonerDensityUsed(77), Integer32 (-2..100)
1394 INTEGER: MULTI-ROW: The toner density used by documents
1395 in this job for devices that can vary toner density levels.
1396 Level 1 is the lowest density and level 100 is the highest
1397 density level. Devices with a smaller range, SHALL map the
1398 1-100 range evenly onto the implemented range.
1399
1400 ++++++
1401 + Job Progress attributes (requested and consumed)
1402 +
1403 + Pairs of these attributes can be used by monitoring
1404 + applications to show an indication of relative progress
1405 + to users. See section 3.4, entitled '**Monitoring Job**
1406 **Progress**'.
1407 ++++++

1408
1409 jobCopiesRequested(90), Integer32 (-2..2147483647)
1410 INTEGER: The number of copies of the entire job that are
1411 to be produced.
1412
1413 jobCopiesCompleted(91), Integer32 (-2..2147483647)
1414 INTEGER: The number of copies of the entire job that have
1415 been completed so far.
1416
1417 documentCopiesRequested(92), Integer32 (-2..2147483647)
1418 INTEGER: The total count of the number of document copies
1419 requested for the job as a whole. If there are documents
1420 A, B, and C, and document B is specified to produce 4
1421 copies, the number of document copies requested is 6 for
1422 the job.
1423
1424 This attribute SHALL be used only when a job has multiple
1425 documents. The jobCopiesRequested attribute SHALL be used
1426 when the job has only one document.
1427
1428 documentCopiesCompleted(93), Integer32 (-2..2147483647)
1429 INTEGER: The total count of the number of document copies
1430 completed so far for the job as a whole. If there are
1431 documents A, B, and C, and document B is specified to
1432 produce 4 copies, the number of document copies starts a 0
1433 and runs up to 6 for the job as the job processes.
1434
1435 This attribute SHALL be used only when a job has multiple
1436 documents. The jobCopiesCompleted attribute SHALL be used
1437 when the job has only one document.
1438

1439 jobKOctetsTransferred(94), Integer32 (-2..2147483647)
1440 INTEGER: The number of K (1024) octets transferred to the
1441 server or device to which the agent is providing access.
1442 This count is independent of the number of copies of the
1443 job or documents that will be produced, but it is only a
1444 measure of the number of bytes transferred to the server or
1445 device.
1446
1447 The agent SHALL round the actual number of octets
1448 transferred up to the next higher K. Thus 0 octets SHALL
1449 be represented as '0', 1-1024 octets SHALL BE represented
1450 as '1', 1025-2048 SHALL be '2', etc. When the job
1451 completes, the values of the jmJobKOctetsPerCopyRequested
1452 object and the jobKOctetsTransferred attribute SHALL be
1453 equal.
1454
1455 NOTE - The jobKOctetsTransferred can be used with the
1456 jmJobKOctetsPerCopyRequested object in order to produce a
1457 relative indication of the progress of the job for agents
1458 that do not implement the jmJobKOctetsProcessed object.
1459
1460 sheetCompletedCopyNumber(95), Integer32 (-2..2147483647)
1461 INTEGER: The number of the copy being stacked for the
1462 current document. This number starts at 0, is set to 1
1463 when the first sheet of the first copy for each document is
1464 being stacked and is equal to n where n is the nth sheet
1465 stacked in the current document copy. See section 3.4 ,
1466 entitled 'Monitoring Job Progress'.
1467
1468 sheetCompletedDocumentNumber(96), Integer32 (-2..2147483647)
1469 INTEGER: The ordinal number of the document in the job
1470 that is currently being stacked. This number starts at 0,
1471 increments to 1 when the first sheet of the first document
1472 in the job is being stacked, and is equal to n where n is
1473 the nth document in the job, starting with 1.
1474
1475 Implementations that only support one document jobs SHOULD
1476 NOT implement this attribute.
1477
1478 jobCollationType(97), JmJobCollationTypeTC
1479 INTEGER: The type of job collation. See also Section 3.4,
1480 entitled 'Monitoring Job Progress'.
1481

```
1482 ++++++
1483 + Impression attributes
1484 +
1485 + See the definition of the terms 'impression', 'sheet',
1486 + and 'page' in Section 2.
1487 +
1488 + See also jmJobImpressionsPerCopyRequested and
1489 + jmJobImpressionsCompleted objects in the jmJobTable.
1490 ++++++
1491
1492 impressionsSpooled(110), Integer32 (-2..2147483647)
1493     INTEGER: The number of impressions spooled to the server
1494     or device for the job so far.
1495
1496 impressionsSentToDevice(111), Integer32 (-2..2147483647)
1497     INTEGER: The number of impressions sent to the device for
1498     the job so far.
1499
1500 impressionsInterpreted(112), Integer32 (-2..2147483647)
1501     INTEGER: The number of impressions interpreted for the job
1502     so far.
1503
1504 impressionsCompletedCurrentCopy(113),
1505     Integer32 (-2..2147483647)
1506     INTEGER: The number of impressions completed by the device
1507     for the current copy of the current document so far. For
1508     printing, the impressions completed includes interpreting,
1509     marking, and stacking the output. For other types of job
1510     services, the number of impressions completed includes the
1511     number of impressions processed.
1512
1513     This value SHALL be reset to 0 for each document in the job
1514     and for each document copy.
1515
1516 fullColorImpressionsCompleted(114), Integer32 (-2..2147483647)
1517     INTEGER: The number of full color impressions completed by
1518     the device for this job so far. For printing, the
1519     impressions completed includes interpreting, marking, and
1520     stacking the output. For other types of job services, the
1521     number of impressions completed includes the number of
1522     impressions processed. Full color impressions are typically
1523     defined as those requiring 3 or more colorants, but this
1524     MAY vary by implementation. In any case, the value of this
1525     attribute counts by 1 for each side that has full color,
1526     not by the number of colors per side (and the other
1527     impression counters are incremented, except
1528     highlightColorImpressionsCompleted(115)).
1529
```

1530 highlightColorImpressionsCompleted(115),
1531 Integer32 (-2..2147483647)
1532 INTEGER: The number of highlight color impressions
1533 completed by the device for this job so far. For printing,
1534 the impressions completed includes interpreting, marking,
1535 and stacking the output. For other types of job services,
1536 the number of impressions completed includes the number of
1537 impressions processed. Highlight color impressions are
1538 typically defined as those requiring black plus one other
1539 colorant, but this MAY vary by implementation. In any
1540 case, the value of this attribute counts by 1 for each side
1541 that has highlight color (and the other impression counters
1542 are incremented, except
1543 fullColorImpressionsCompleted(114)).
1544
1545 ++++++
1546 + Page attributes
1547 +
1548 + See the definition of 'impression', 'sheet', and 'page'
1549 + in Section 2.
1550 ++++++

1551
1552 pagesRequested(130), Integer32 (-2..2147483647)
1553 INTEGER: The number of logical pages requested by the job
1554 to be processed.
1555

1556 pagesCompleted(131), Integer32 (-2..2147483647)
1557 INTEGER: The number of logical pages completed for this
1558 job so far.
1559
1560 For implementations where multiple copies are produced by
1561 the interpreter with only a single pass over the data, the
1562 final value SHALL be equal to the value of the
1563 pagesRequested object. For implementations where multiple
1564 copies are produced by the interpreter by processing the
1565 data for each copy, the final value SHALL be a multiple of
1566 the value of the pagesRequested object.
1567

1568 NOTE - See the impressionsCompletedCurrentCopy and
1569 pagesCompletedCurrentCopy attributes for attributes that
1570 are reset on each document copy.
1571

1572 NOTE - The pagesCompleted object can be used with the
1573 pagesRequested object to provide an indication of the
1574 relative progress of the job, provided that the
1575 multiplicative factor is taken into account for some
1576 implementations of multiple copies.
1577

1578 pagesCompletedCurrentCopy(132), Integer32 (-2..2147483647)
1579 INTEGER: The number of logical pages completed for the
1580 current copy of the document so far. This value SHALL be
1581 reset to 0 for each document in the job and for each
1582 document copy.
1583
1584 ++++++
1585 + Sheet attributes
1586 +
1587 + See the definition of 'impression', 'sheet', and 'page'
1588 + in Section 2.
1589 ++++++

1591 sheetsRequested(150), Integer32 (-2..2147483647)
1592 INTEGER: The total number of medium sheets requested to be
1593 produced for this job.
1594
1595 Unlike the jmJobKOctetsPerCopyRequested and
1596 jmJobImpressionsPerCopyRequested attributes, the
1597 sheetsRequested(150) attribute SHALL include the
1598 multiplicative factor contributed by the number of copies
1599 and so is the total number of sheets to be produced by the
1600 job, as opposed to the size of the document(s) submitted.
1601

1602 sheetsCompleted(151), Integer32 (-2..2147483647)
1603 INTEGER: The total number of medium sheets that have
1604 completed marking and stacking for the entire job so far
1605 whether those sheets have been processed on one side or on
1606 both.
1607

1608 sheetsCompletedCurrentCopy(152), Integer32 (-2..2147483647)
1609 INTEGER: The number of medium sheets that have completed
1610 marking and stacking for the current copy of a document in
1611 the job so far whether those sheets have been processed on
1612 one side or on both.
1613
1614 The value of this attribute SHALL be 0 before the job
1615 starts processing and SHALL be reset to 1 after the first
1616 sheet of each document and document copy in the job is
1617 processed and stacked.
1618


```

1619 ++++++
1620 + Resources attributes (requested and consumed)
1621 +
1622 + Pairs of these attributes can be used by monitoring
1623 + applications to show an indication of relative usage to
1624 + users, i.e., a 'thermometer'.
1625 ++++++
1626
1627 mediumRequested(170),                               JmMediumTypeTC
1628                                                         AND/OR
1629                                                         JmJobStringTC (SIZE(0..63))
1630     INTEGER:  MULTI-ROW:  The type
1631     AND/OR
1632     OCTETS:  MULTI-ROW:  the name of the medium that is
1633     required by the job.
1634
1635     NOTE - The name (JmJobStringTC) values correspond to the
1636     name values of the prtInputMediaName object in the Printer
1637     MIB [print-mib] and the name, size, and input tray values
1638     of the IPP 'media' attribute [ipp-model].
1639
1640 mediumConsumed(171),                               Integer32 (-2..2147483647)
1641                                                         AND
1642                                                         JmJobStringTC (SIZE(0..63))
1643     INTEGER:  MULTI-ROW:  The number of sheets
1644     AND
1645     OCTETS:  MULTI-ROW:  the name of the medium that has been
1646     consumed so far whether those sheets have been processed on
1647     one side or on both.
1648
1649     This attribute SHALL have both Integer32 and OCTET STRING
1650     (represented as JmJobStringTC) values.
1651
1652     NOTE - The name (JmJobStringTC) values correspond to the
1653     name values of the prtInputMediaName object in the Printer
1654     MIB [print-mib] and the name, size, and input tray values
1655     of the IPP 'media' attribute [ipp-model].
1656
1657 colorantRequested(172),                             Integer32 (-2..2147483647)
1658                                                         AND/OR
1659                                                         JmJobStringTC (SIZE(0..63))
1660     INTEGER:  MULTI-ROW:  The index (prtMarkerColorantIndex) in
1661     the Printer MIB[print-mib]
1662     AND/OR
1663     OCTETS:  MULTI-ROW:  the name of the colorant requested.
1664
1665     NOTE - The name (JmJobStringTC) values correspond to the
1666     name values of the prtMarkerColorantValue object in the
1667     Printer MIB.  Examples are: red, blue.

```

1668
1669 colorantConsumed(173), Integer32 (-2..2147483647)
1670 AND/OR
1671 JmJobStringTC (SIZE(0..63))
1672 INTEGER: MULTI-ROW: The index (prtMarkerColorantIndex) in
1673 the Printer MIB[print-mib]
1674 AND/OR
1675 OCTETS: MULTI-ROW: the name of the colorant consumed.
1676
1677 NOTE - The name (JmJobStringTC) values correspond to the
1678 name values of the prtMarkerColorantValue object in the
1679 Printer MIB. Examples are: red, blue
1680
1681 mediumTypeConsumed(174), Integer32 (-2..2147483647)
1682 AND
1683 JmJobStringTC (SIZE(0..63))
1684 INTEGER: MULTI-ROW: The number of sheets of the indicated
1685 medium type that has been consumed so far whether those
1686 sheets have been processed on one side or on both
1687 AND
1688 OCTETS: MULTI-ROW: the name of that medium type.
1689
1690 This attribute SHALL have both Integer32 and OCTET STRING
1691 (represented as JmJobStringTC) values.
1692
1693 NOTE - The type name (JmJobStringTC) values correspond to
1694 the type name values of the prtInputMediaType object in the
1695 Printer MIB [print-mib]. Values are: 'stationery',
1696 'transparency', 'envelope', etc. These medium type names
1697 correspond to the enum values of JmMediumTypeTC used in the
1698 mediumRequested attribute.
1699
1700 mediumSizeConsumed(175), Integer32 (-2..2147483647)
1701 AND
1702 JmJobStringTC (SIZE(0..63))
1703 INTEGER: MULTI-ROW: The number of sheets of the indicated
1704 medium size that has been consumed so far whether those
1705 sheets have been processed on one side or on both
1706 AND
1707 OCTETS: MULTI-ROW: the name of that medium size.
1708
1709 This attribute SHALL have both Integer32 and OCTET STRING
1710 (represented as JmJobStringTC) values.
1711
1712 NOTE - The size name (JmJobStringTC) values correspond to
1713 the size name values in the Printer MIB [print-mib]
1714 Appendix B. These size name values are also a subset of
1715 the keyword values defined by [ipp-model] for the 'media'
1716 Job Template attribute. Values are: 'letter', 'a', 'iso-
1717 a4', 'jis-b4', etc.
1718

```

1719 ++++++
1720 + Time attributes (set by server or device)
1721 +
1722 + This section of attributes are ones that are set by the
1723 + server or device that accepts jobs. Two forms of time are
1724 + provided. Each form is represented in a separate attribute.
1725 + See section 3.1.2 and section 3.1.3 for the
1726 + conformance requirements for time attribute for agents and
1727 + monitoring applications, respectively. The two forms are:
1728 +
1729 + 'DateAndTime' is an 8 or 11 octet binary encoded year,
1730 + month, day, hour, minute, second, deci-second with
1731 + optional offset from UTC. See SNMPv2-TC [SMIV2-TC].
1732 +
1733 + NOTE: 'DateAndTime' is not printable characters; it is
1734 + binary.
1735 +
1736 + 'JmTimeStampTC' is the time of day measured in the number of
1737 + seconds since the system was booted.
1738 ++++++
1739
1740 jobSubmissionToServerTime(190),      JmTimeStampTC
1741                                     AND/OR
1742                                     DateAndTime
1743     INTEGER: Configuration 3 only: The time
1744     AND/OR
1745     OCTETS:  the date and time that the job was submitted to
1746     the server (as distinguished from the device which uses
1747     jobSubmissionTime).
1748
1749 jobSubmissionTime(191),              JmTimeStampTC
1750                                     AND/OR
1751                                     DateAndTime
1752     INTEGER: Configurations 1, 2, and 3: The time
1753     AND/OR
1754     OCTETS:  the date and time that the job was submitted to
1755     the server or device to which the agent is providing
1756     access.
1757
1758 jobStartedBeingHeldTime(192),        JmTimeStampTC
1759                                     AND/OR
1760                                     DateAndTime
1761     INTEGER: The time
1762     AND/OR
1763     OCTETS:  the date and time that the job last entered the
1764     pendingHeld state. If the job has never entered the
1765     pendingHeld state, then the value SHALL be '0' or the
1766     attribute SHALL not be present in the table.

```

1767
1768 jobStartedProcessingTime(193), JmTimeStampTC
1769 AND/OR
1770 DateAndTime
1771 INTEGER: The time
1772 AND/OR
1773 OCTETS: the date and time that the job started processing.
1774
1775 jobCompletionTime(194), JmTimeStampTC
1776 AND/OR
1777 DateAndTime
1778 INTEGER: The time
1779 AND/OR
1780 OCTETS: the date and time that the job entered the
1781 completed, canceled, or aborted state.
1782
1783 jobProcessingCPUtime(195) Integer32 (-2..2147483647)
1784 UNITS 'seconds'
1785 INTEGER: The amount of CPU time in seconds that the job
1786 has been in the processing state. If the job enters the
1787 processingStopped state, that elapsed time SHALL not be
1788 included. In other words, the jobProcessingCPUtime value
1789 SHOULD be relatively repeatable when the same job is
1790 processed again on the same device.

1791 3.3.9 Job State Reason bit definitions

1792 The JmJobStateReasonsMTC ($N=1..4$) textual-conventions are used with the
1793 jmJobStateReasons1 object and jobStateReasonsN ($N=2..4$), respectively,
1794 to provide additional information regarding the current jmJobState
1795 object value. These values MAY be used with any job state or states
1796 for which the reason makes sense.

1797 NOTE - While values cannot be added to the jmJobState object without
1798 impacting deployed clients that take actions upon receiving jmJobState
1799 values, it is the intent that additional JmJobStateReasonsMTC enums can
1800 be defined and registered without impacting such deployed clients. In
1801 other words, the jmJobStateReasons1 object and jobStateReasonsN
1802 attributes are intended to be extensible.

1803 NOTE - The Job Monitoring MIB contains a superset of the IPP
1804 values[ipp-model] for the IPP 'job-state-reasons' attribute, since the
1805 Job Monitoring MIB is intended to cover other job submission protocols
1806 as well. Also some of the names of the reasons have been changed from
1807 'printer' to 'device', since the Job Monitoring MIB is intended to
1808 cover additional types of devices, including input devices, such as
1809 scanners.

1810 **3.3.9.1 JmJobStateReasons1TC specification**

1811 The following standard values are defined (in hexadecimal) as *powers of*
1812 *two*, since multiple values MAY be used at the same time. For ease of
1813 understanding, the JmJobStateReasons1TC reasons are presented in the
1814 order in which the reasons are likely to occur (if implemented),
1815 starting with the 'jobIncoming' value and ending with the
1816 'jobCompletedWithErrors' value.

1817
1818 other 0x1
1819 The job state reason is not one of the standardized or
1820 registered reasons.
1821
1822 unknown 0x2
1823 The job state reason is not known to the agent or is
1824 indeterminent.
1825
1826 jobIncoming 0x4
1827 The job has been accepted by the server or device, but the
1828 server or device is expecting (1) additional operations
1829 from the client to finish creating the job and/or (2) is
1830 accessing/accepting document data.
1831
1832 submissionInterrupted 0x8
1833 The job was not completely submitted for some unforeseen
1834 reason, such as: (1) the server has crashed before the job
1835 was closed by the client, (2) the server or the document
1836 transfer method has crashed in some non-recoverable way
1837 before the document data was entirely transferred to the
1838 server, (3) the client crashed or failed to close the job
1839 before the time-out period.
1840
1841 jobOutgoing 0x10
1842 Configuration 2 only: The server is transmitting the job
1843 to the device.
1844
1845 jobHoldSpecified 0x20
1846 The value of the job's jobHold(52) attribute is TRUE. The
1847 job SHALL NOT be a candidate for processing until this
1848 reason is removed and there are no other reasons to hold
1849 the job.
1850
1851 jobHoldUntilSpecified 0x40
1852 The value of the job's jobHoldUntil(53) attribute specifies
1853 a time period that is still in the future. The job SHALL
1854 NOT be a candidate for processing until this reason is
1855 removed and there are no other reasons to hold the job.
1856
1857 jobProcessAfterSpecified 0x80
1858 The value of the job's jobProcessAfterDateAndTime(51)
1859 attribute specifies a time that is still in the future.

1860 The job SHALL NOT be a candidate for processing until this
1861 reason is removed and there are no other reasons to hold
1862 the job.
1863

1864 resourcesAreNotReady 0x100
1865 At least one of the resources needed by the job, such as
1866 media, fonts, resource objects, etc., is not ready on any
1867 of the physical devices for which the job is a candidate.
1868 This condition MAY be detected when the job is accepted, or
1869 subsequently while the job is pending or processing,
1870 depending on implementation.
1871

1872 deviceStoppedPartly 0x200
1873 One or more, but not all, of the devices to which the job
1874 is assigned are stopped. If all of the devices are stopped
1875 (or the only device is stopped), the deviceStopped reason
1876 SHALL be used.
1877

1878 deviceStopped 0x400
1879 The device(s) to which the job is assigned is (are all)
1880 stopped.
1881

1882 jobInterpreting 0x800
1883 The device to which the job is assigned is interpreting the
1884 document data.
1885

1886 jobPrinting 0x1000
1887 The output device to which the job is assigned is marking
1888 media. This value is useful for servers and output devices
1889 which spend a great deal of time processing (1) when no
1890 marking is happening and then want to show that marking is
1891 now happening or (2) when the job is in the process of
1892 being canceled or aborted while the job remains in the
1893 processing state, but the marking has not yet stopped so
1894 that impression or sheet counts are still increasing for
1895 the job.
1896

1897 jobCanceledByUser 0x2000
1898 The job was canceled by the owner of the job, i.e., by a
1899 user whose name is the same as the value of the job's
1900 jmJobOwner object, or by some other authorized end-user,
1901 such as a member of the job owner's security group.
1902

1903 jobCanceledByOperator 0x4000
1904 The job was canceled by the operator, i.e., by a user who
1905 has been authenticated as having operator privileges
1906 (whether local or remote).
1907

1908 jobCanceledAtDevice 0x8000
1909 The job was canceled by an unidentified local user, i.e., a
1910 user at a console at the device.
1911

1912 abortedBySystem 0x10000
1913 The job (1) is in the process of being aborted, (2) has
1914 been aborted by the system and placed in the 'aborted'
1915 state, or (3) has been aborted by the system and placed in
1916 the 'pendingHeld' state, so that a user or operator can
1917 manually try the job again.
1918
1919 processingToStopPoint 0x20000
1920 The requester has issued an operation to cancel or
1921 interrupt the job or the server/device has aborted the job,
1922 but the server/device is still performing some actions on
1923 the job until a specified stop point occurs or job
1924 termination/cleanup is completed.
1925
1926 This reason is recommended to be used in conjunction with
1927 the processing job state to indicate that the server/device
1928 is still performing some actions on the job while the job
1929 remains in the processing state. After all the job's
1930 resources consumed counters have stopped incrementing, the
1931 server/device moves the job from the processing state to
1932 the canceled or aborted job states.
1933
1934 serviceOffLine 0x40000
1935 The service or document transform is off-line and accepting
1936 no jobs. All pending jobs are put into the pendingHeld
1937 state. This situation could be true if the service's or
1938 document transform's input is impaired or broken.
1939
1940 jobCompletedSuccessfully 0x80000
1941 The job completed successfully.
1942
1943 jobCompletedWithWarnings 0x100000
1944 The job completed with warnings.
1945
1946 jobCompletedWithErrors 0x200000
1947 The job completed with errors (and possibly warnings too).
1948
1949 The following additional job state reasons have been added to represent
1950 job states that are in ISO DPA[iso-dpa] and other job submission
1951 protocols:
1952
1953 jobPaused 0x400000
1954 The job has been indefinitely suspended by a client issuing
1955 an operation to suspend the job so that other jobs may
1956 proceed using the same devices. The client MAY issue an
1957 operation to resume the paused job at any time, in which
1958 case the agent SHALL remove the jobPaused values from the
1959 job's jmJobStateReasons1 object and the job is eventually
1960 resumed at or near the point where the job was paused.
1961

1962 jobInterrupted 0x800000
 1963 The job has been interrupted while processing by a client
 1964 issuing an operation that specifies another job to be run
 1965 instead of the current job. The server or device will
 1966 automatically resume the interrupted job when the
 1967 interrupting job completes.
 1968
 1969 jobRetained 0x1000000
 1970 The job is being retained by the server or device with all
 1971 of the job's document data (and submitted resources, such
 1972 as fonts, logos, and forms, if any). Thus a client could
 1973 issue an operation to the server or device to either (1)
 1974 re-do the job (or a copy of the job) on the same server or
 1975 device or (2) resubmit the job to another server or device.
 1976 When a client could no longer re-do/resubmit the job, such
 1977 as after the document data has been discarded, the agent
 1978 SHALL remove the jobRetained value from the
 1979 jmJobStateReasons1 object.
 1980

1981 These bit definitions are the equivalent of a type 2 enum except that
 1982 combinations of bits may be used together. See section 3.7.1.2. The
 1983 remaining bits are reserved for future standardization and/or
 1984 registration.

1985 3.3.9.2 JmJobStateReasons2TC specification

1986 The following standard values are defined (in hexadecimal) as *powers of*
 1987 *two*, since multiple values MAY be used at the same time.

1988
 1989 cascaded 0x1
 1990 An outbound gateway has transmitted all of the job's job
 1991 and document attributes and data to another spooling
 1992 system.
 1993
 1994 deletedByAdministrator 0x2
 1995 The administrator has deleted the job.
 1996
 1997 discardTimeArrived 0x4
 1998 The job has been deleted due to the fact that the time
 1999 specified by the job's job-discard-time attribute has
 2000 arrived.
 2001
 2002 postProcessingFailed 0x8
 2003 The post-processing agent failed while trying to log
 2004 accounting attributes for the job; therefore the job has
 2005 been placed into the completed state with the jobRetained
 2006 jmJobStateReasons1 object value for a system-defined period
 2007 of time, so the administrator can examine it, resubmit it,
 2008 etc.
 2009

2010 jobTransforming 0x10
2011 The server/device is interpreting document data and
2012 producing another electronic representation.
2013

2014 maxJobFaultCountExceeded 0x20
2015 The job has faulted several times and has exceeded the
2016 administratively defined fault count limit.
2017

2018 devicesNeedAttentionTimeOut 0x40
2019 One or more document transforms that the job is using needs
2020 human intervention in order for the job to make progress,
2021 but the human intervention did not occur within the site-
2022 settable time-out value.
2023

2024 needsKeyOperatorTimeOut 0x80
2025 One or more devices or document transforms that the job is
2026 using need a specially trained operator (who may need a key
2027 to unlock the device and gain access) in order for the job
2028 to make progress, but the key operator intervention did not
2029 occur within the site-settable time-out value.
2030

2031 jobStartWaitTimeOut 0x100
2032 The server/device has stopped the job at the beginning of
2033 processing to await human action, such as installing a
2034 special cartridge or special non-standard media, but the
2035 job was not resumed within the site-settable time-out value
2036 and the server/device has transitioned the job to the
2037 pendingHeld state.
2038

2039 jobEndWaitTimeOut 0x200
2040 The server/device has stopped the job at the end of
2041 processing to await human action, such as removing a
2042 special cartridge or restoring standard media, but the job
2043 was not resumed within the site-settable time-out value and
2044 the server/device has transitioned the job to the completed
2045 state.
2046

2047 jobPasswordWaitTimeOut 0x400
2048 The server/device has stopped the job at the beginning of
2049 processing to await input of the job's password, but the
2050 password was not received within the site-settable time-out
2051 value.
2052

2053 deviceTimedOut 0x800
2054 A device that the job was using has not responded in a
2055 period specified by the device's site-settable attribute.
2056

2057 connectingToDeviceTimeOut 0x1000
2058 The server is attempting to connect to one or more devices
2059 which may be dial-up, polled, or queued, and so may be busy
2060 with traffic from other systems, but server was unable to

2061 connect to the device within the site-settable time-out
2062 value.
2063
2064 transferring 0x2000
2065 The job is being transferred to a down stream server or
2066 downstream device.
2067
2068 queuedInDevice 0x4000
2069 The server/device has queued the job in a down stream
2070 server or downstream device.
2071
2072 jobQueued 0x8000
2073 The server/device has queued the document data.
2074
2075 jobCleanup 0x10000
2076 The server/device is performing cleanup activity as part of
2077 ending normal processing.
2078
2079 jobPasswordWait 0x20000
2080 The server/device has selected the job to be next to
2081 process, but instead of assigning resources and starting
2082 the job processing, the server/device has transitioned the
2083 job to the pendingHeld state to await entry of a password
2084 (and dispatched another job, if there is one).
2085
2086 validating 0x40000
2087 The server/device is validating the job *after* accepting the
2088 job.
2089
2090 queueHeld 0x80000
2091 The operator has held the entire job set or queue.
2092
2093 jobProofWait 0x100000
2094 The job has produced a single proof copy and is in the
2095 pendingHeld state waiting for the requester to issue an
2096 operation to release the job to print normally, obeying any
2097 job and document copy attributes that were originally
2098 submitted.
2099
2100 heldForDiagnostics 0x200000
2101 The system is running intrusive diagnostics, so that all
2102 jobs are being held.

2103 noSpaceOnServer 0x800000
2104 There is no room on the server to store all of the job.
2105
2106 pinRequired 0x1000000
2107 The System Administrator settable device policy is (1) to
2108 require PINs, and (2) to hold jobs that do not have a pin
2109 supplied as an input parameter when the job was created.
2110
2111 exceededAccountLimit 0x2000000
2112 The account for which this job is drawn has exceeded its
2113 limit. This condition SHOULD be detected before the job is
2114 scheduled so that the user does not wait until his/her job
2115 is scheduled only to find that the account is overdrawn.
2116 This condition MAY also occur while the job is processing
2117 either as processing begins or part way through processing.
2118
2119 heldForRetry 0x4000000
2120 The job encountered some errors that the server/device
2121 could not recover from with its normal retry procedures,
2122 but the error might not be encountered if the job is
2123 processed again in the future. Example cases are phone
2124 number busy or remote file system in-accessible. For such
2125 a situation, the server/device SHALL transition the job
2126 from the processing to the pendingHeld, rather than to the
2127 aborted state.
2128
2129 The following values are from the X/Open PSIS draft standard:
2130
2131 canceledByShutdown 0x8000000
2132 The job was canceled because the server or device was
2133 shutdown before completing the job.
2134
2135 deviceUnavailable 0x10000000
2136 This job was aborted by the system because the device is
2137 currently unable to accept jobs.
2138
2139 wrongDevice 0x20000000
2140 This job was aborted by the system because the device is
2141 unable to handle this particular job; the spooler SHOULD
2142 try another device or the user should submit the job to
2143 another device.
2144
2145 badJob 0x40000000
2146 This job was aborted by the system because this job has a
2147 major problem, such as an ill-formed PDL; the spooler
2148 SHOULD not even try another device.
2149

2150 These bit definitions are the equivalent of a type 2 enum except that
2151 combinations of them may be used together. See section 3.7.1.2.

2152 3.3.9.3 JmJobStateReasons3TC specification

2153 This textual-convention is used with the jobStateReasons3 attribute to
2154 provides additional information regarding the jmJobState object. The
2155 following standard values are defined (in hexadecimal) as *powers of*
2156 *two*, since multiple values may be used at the same time:

2157
2158 jobInterruptedByDeviceFailure 0x1
2159 A device or the print system software that the job was
2160 using has failed while the job was processing. The server
2161 or device is keeping the job in the pendingHeld state until
2162 an operator can determine what to do with the job.

2163 These bit definitions are the equivalent of a type 2 enum except that
2164 combinations of them may be used together. See section 3.7.1.2. The
2165 remaining bits are reserved for future standardization and/or
2166 registration.

2167 3.3.9.4 JmJobStateReasons4TC specification

2168 This textual-convention is used with the jobStateReasons4 attribute to
2169 provides additional information regarding the jmJobState object. The
2170 following standard values are defined (in hexadecimal) as *powers of*
2171 *two*, since multiple values MAY be used at the same time.

2172
2173 None defined at this time.

2174 These bit definitions are the equivalent of a type 2 enum except that
2175 combinations of them may be used together. See section 3.7.1.2. The
2176 remaining bits are reserved for future standardization and/or
2177 registration.

2178 3.4 Monitoring Job Progress

2179 There are a number of objects and attributes for monitoring the
2180 progress of a job. These objects and attributes count the number of K
2181 octets, impressions, sheets, and pages requested or completed. For
2182 impressions and sheets, "completed" means stacked, unless the
2183 implementation is unable to detect when each sheet is stacked, in which
2184 case stacked is approximated when processing of each sheet completes.
2185 There are objects and attributes for the overall job and for the
2186 current copy of the document currently being stacked. For the latter,
2187 the rate at which the various objects and attributes count depends on
2188 the sheet and document collation of the job.

2189 Job Collation included sheet collation and document collation. Sheet
2190 collation is defined to be the ordering of sheets within a document
2191 copy. Document collation is defined to be ordering of document copies
2192 within a multi-document job. There are three types of job collation
2193 (see terminology definitions in Section 2):

2194 1. uncollatedSheets(3) - No collation of the sheets within each
2195 document copy, i.e., each sheet of a document that is to
2196 produce multiple copies is replicated before the next sheet in
2197 the document is processed and stacked. If the device has an
2198 output bin collator, the uncollatedSheets(3) value may actually
2199 produce collated sheets as far as the user is concerned (in the
2200 output bins). However, when the job collation is the
2201 'uncollatedSheets(3)' value, job progress is indistinguishable
2202 to a monitoring application between a device that has an output
2203 bin collator and one that does not.

2204 2. collatedDocuments(4) - Collation of the sheets within each
2205 document copy is performed within the printing device by making
2206 multiple passes over either the source or an intermediate
2207 representation of the document. In addition, when there are
2208 multiple documents per job, the i'th copy of each document is
2209 stacked before the j'th copy of each document, i.e., the
2210 documents are collated within each job copy. For example, if a
2211 job is submitted with documents, A and B, the job is made
2212 available to the end user as: A, B, A, B, The
2213 'collatedDocuments(4)' value corresponds to the IPP [ipp-model]
2214 'separate-documents-collated-copies' value of the "multiple-
2215 document-handling" attribute.
2216

2217 If jobCopiesRequested or documentCopiesRequested = 1, then
2218 jobCollationType is defined as 4.

2219 3. uncollatedDocuments(5) - Collation of the sheets within each
2220 document copy is performed within the printing device by making
2221 multiple passes over either the source or an intermediate
2222 representation of the document. In addition, when there are
2223 multiple documents per job, all copies of the first document in
2224 the job are stacked before the any copied of the next document
2225 in the job, i.e., the documents are uncollated within the job.
2226 For example, if a job is submitted with documents, A and B, the
2227 job is mad available to the end user as: A, A, ..., B, B,
2228 The 'uncollatedDocuments(5)' value corresponds to the IPP [ipp-
2229 model] 'separate-documents-uncollated-copies' value of the
2230 "multiple-document-handling" attribute.

2231 Consider the following four variables that are used to monitor the
2232 progress of a job's impressions:

2233 1. jmJobImpressionsCompleted - counts the total number of
2234 impressions stacked for the job

2235 2. impressionsCompletedCurrentCopy - counts the number of
2236 impressions stacked for the current document copy

2237 3. sheetCompletedCopyNumber - identifies the number of the copy
2238 for the current document being stacked where the first copy is
2239 1.

2240 4. sheetCompletedDocumentNumber - identifies the current document
2241 within the job that is being stacked where the first document
2242 in a job is 1. NOTE: this attribute SHOULD NOT be implemented
2243 for implementations that only support one document per job.

2244 For each of the three types of job collation, a job with three copies
2245 of two documents (1, 2), where each document consists of 3 impressions,
2246 the four variables have the following values as each sheet is stacked
2247 for one-sided printing:

2248 Job Collation Type = uncollatedSheets(3)

2249

jmJobImpressions Completed	Impressions CompletedCurrent Copy	sheetCompleted CopyNumber	sheetCompleted DocumentNumber
0	0	0	0
1	1	1	1
2	1	2	1
3	1	3	1
4	2	1	1
5	2	2	1
6	2	3	1
7	3	1	1
8	3	2	1
9	3	3	1
10	1	1	2
11	1	2	2
12	1	3	2
13	2	1	2
14	2	2	2
15	2	3	2
16	3	1	2
17	3	2	2
18	3	3	2

2250

2251 Job Collation Type = collatedDocuments(4)

2252

JmJobImpressions Completed	Impressions CompletedCurrent Copy	sheetCompleted CopyNumber	sheetCompleted DocumentNumber
0	0	0	0
1	1	1	1
2	2	1	1
3	3	1	1
4	1	1	2
5	2	1	2
6	3	1	2
7	1	2	1
8	2	2	1
9	3	2	1
10	1	2	2
11	2	2	2
12	3	2	2
13	1	3	1
14	2	3	1
15	3	3	1
16	1	3	2
17	2	3	2
18	3	3	2

2253

2254 Job Collation Type = uncollatedDocuments(5)
 2255

jmJobImpressions Completed	Impressions CompletedCurrent Copy	sheetCompleted CopyNumber	sheetCompleted DocumentNumber
0	0	0	0
1	1	1	1
2	2	1	1
3	3	1	1
4	1	2	1
5	2	2	1
6	3	2	1
7	1	3	1
8	2	3	1
9	3	3	1
10	1	1	2
11	2	1	2
12	3	1	2
13	1	2	2
14	2	2	2
15	3	2	2
16	1	3	2
17	2	3	2
18	3	3	2

2256

2257 3.5 Job Identification

2258 There are a number of attributes that permit a user, operator or system
 2259 administrator to identify jobs of interest, such as jobURI, jobName,
 2260 jobOriginatingHost, etc. In addition, there is a jmJobSubmissionID
 2261 object that is a text string table index. Being a table index allows a
 2262 monitoring application to quickly locate and identify a particular job
 2263 of interest that was submitted from a particular client by the user
 2264 invoking the monitoring application without having to scan the entire
 2265 job table. The Job Monitoring MIB needs to provide for identification
 2266 of the job at both sides of the job submission process. The primary
 2267 identification point is the client side. The jmJobSubmissionID allows
 2268 the monitoring application to identify the job of interest from all the
 2269 jobs currently "known" by the server or device. The value of
 2270 jmJobSubmissionID can be assigned by either the client's local system
 2271 or a downstream server or device. The point of assignment depends on
 2272 the job submission protocol in use.

2273 The server/device-side identifier, called the jmJobIndex object, SHALL
 2274 be assigned by the SNMP Job Monitoring MIB agent when the server or
 2275 device accepts the jobs from submitting clients. The jmJobIndex object
 2276 allows the interested party to obtain all objects desired that relate
 2277 to a particular job. See Section 3.2, entitled 'The Job Tables and the

2278 Oldest Active and Newest Active Indexes' for the specification of how
2279 the agent SHALL assign the jmJobIndex values.

2280 The MIB provides a mapping table that maps each jmJobSubmissionID value
2281 to a corresponding jmJobIndex value generated by the agent, so that an
2282 application can determine the correct value for the jmJobIndex value
2283 for the job of interest in a single Get operation, given the Job
2284 Submission ID. See the jmJobIDGroup.

2285 In some configurations there may be more than one application program
2286 that monitors the same job when the job passes from one network entity
2287 to another when it is submitted. See configuration 3. When there are
2288 multiple job submission IDs, each entity MAY supply an appropriate
2289 jmJobSubmissionID value. In this case there would be a separate entry
2290 in the jmJobSubmissionID table, one for each jmJobSubmissionID. All
2291 entries would map to the same jmJobIndex that contains the job data.
2292 When the job is deleted, it is up to the agent to remove all entries
2293 that point to the job from the jmJobSubmissionID table as well.

2294 The jobName attribute provides a name that the user supplies as a job
2295 attribute with the job. The jobName attribute is not necessarily
2296 unique, even for one user, let alone across users.

2297 **3.5.1 The Job Submission ID specifications**

2298 This section specifies the formats for each of the registered Job
2299 Submission Ids. This format is used by the JmJobSubmissionIDTypeTC.
2300 Each job submission ID is a fixed-length, 48-octet printable US-ASCII
2301 [US-ASCII] coded character string containing no control characters,
2302 consisting of the following fields:

2303

2304 octet 1: The format letter identifying the format. The US-
2305 ASCII characters '0-9', 'A-Z', and 'a-z' are assigned in
2306 order giving 62 possible formats.

2307 octets 2-40: A 39-character, US-ASCII trailing SPACE filled
2308 field specified by the format letter, if the data is less
2309 than 39 ASCII characters.

2310 octets 41-48: A sequential or random US-ASCII number to make
2311 the ID quasi-unique.

2312

2313 If the client does not supply a job submission ID in the job submission
2314 protocol, then the agent SHALL assign a job submission ID using any of
2315 the standard formats that are reserved for the agent. Clients SHALL
2316 not use formats that are reserved for agents and agents SHALL NOT use
2317 formats that are reserved for clients, in order to reduce conflicts in
2318 ID generation. See the description for which formats are reserved for
2319 clients or for agents.

2320 Registration of additional formats may be done following the procedures
2321 described in Section 3.7.3.

2322 The format values defined at the time of completion of this
2323 specification are:

2324
2325 Format
2326 Letter Description
2327 ----- -----
2328 '0' Job Owner generated by the server/device
2329 octets 2-40: The last 39 bytes of the jmJobOwner object.
2330 octets 41-48: The US-ASCII 8-decimal-digit sequential number
2331 assigned by the agent.
2332 This format is reserved for agents.
2333
2334 NOTE - Clients wishing to use a job submission ID that
2335 incorporates the job owner, SHALL use format '8', not
2336 format '0'.
2337
2338 '1' Job Name
2339 octets 2-40: The last 39 bytes of the jobName attribute.
2340 octets 41-48: The US-ASCII 8-decimal-digit random number
2341 assigned by the client.
2342 This format is reserved for clients.
2343
2344 '2' Client MAC address
2345 octets 2-40: The client MAC address: in hexadecimal with each
2346 nibble of the 6 octet address being '0'-'9' or 'A' - 'F'
2347 (uppercase only). Most significant octet first.
2348 octets 41-48: The US-ASCII 8-decimal-digit sequential number
2349 assigned by the client.
2350 This format is reserved for clients.
2351
2352 '3' Client URL
2353 octets 2-40: The last 39 bytes of the client URL [URI-spec].
2354 octets 41-48: The US-ASCII 8-decimal-digit sequential number
2355 assigned by the client.
2356 This format is reserved for clients.
2357
2358 '4' Job URI
2359 octets 2-40: The last 39 bytes of the URI [URI-spec] assigned
2360 by the server or device to the job when the job was
2361 submitted for processing.
2362 octets 41-48: The US-ASCII 8-decimal-digit sequential number
2363 assigned by the agent.
2364 This format is reserved for agents.
2365
2366 '5' POSIX User Number
2367 octets 2-40: The last 39 bytes of a user number, such as POSIX
2368 user number.
2369 octets 41-48: The US-ASCII 8-decimal-digit sequential number
2370 assigned by the client.

2371 This format is reserved for clients.
2372
2373 '6' User Account Number
2374 octets 2-40: The last 39 bytes of the user account number.
2375 octets 41-48: The US-ASCII 8-decimal-digit sequential number
2376 assigned by the client.
2377 This format is reserved for clients.
2378
2379 '7' DTMF Incoming FAX routing number
2380 octets 2-40: The last 39 bytes of the DTMF incoming FAX
2381 routing number.
2382 octets 41-48: The US-ASCII 8-decimal-digit sequential number
2383 assigned by the client.
2384 This format is reserved for clients.
2385
2386 '8' Job Owner supplied by the client
2387 octets 2-40: The last 39 bytes of the job owner name (that the
2388 agent returns in the jmJobOwner object).
2389 octets 41-48: The US-ASCII 8-decimal-digit sequential number
2390 assigned by the client.
2391 This format is reserved for clients. See format '0' which is
2392 reserved for agents.
2393
2394 '9' Host Name
2395 octets 2-40: The last 39 bytes of the host name with trailing
2396 SPACES that submitted the job to this server/device using a
2397 protocol, such as LPD [RFC-1179] which includes the host
2398 name in the job submission protocol.
2399 octets 41-48: The US-ASCII 8-decimal-digit leading zero
2400 representation of the job id generated by the submitting
2401 server (configuration 3) or the client (configuration 1 and
2402 2), such as in the LPD protocol.
2403 This format is reserved for clients.
2404
2405 'A' AppleTalk Protocol
2406 octets 2-40: Contains the AppleTalk printer name, with the
2407 first character of the name in octet 2. AppleTalk printer
2408 names are a maximum of 31 characters. Any unused portion
2409 of this field shall be filled with spaces.
2410 octets 41-48: '00000XXX', where 'XXX' is the 3-digit US-ASCII
2411 decimal representation of the Connection Id.
2412 This format is reserved for agents.
2413

2414 'B' NetWare PServer
2415 octets 2-40: Contains the Directory Path Name as recorded by
2416 the Novell File Server in the queue directory. If the
2417 string is less than 40 octets, the left-most character in
2418 the string shall appear in octet position 2. Otherwise,
2419 only the last 39 bytes shall be included. Any unused
2420 portion of this field shall be filled with spaces.
2421 octets 41-48: '000XXXXX' The US-ASCII representation of the
2422 Job Number as per the NetWare File Server Queue Management
2423 Services.
2424 This format is reserved for agents.
2425
2426 'C' Server Message Block protocol (SMB)
2427 octets 2-40: Contains a decimal (US-ASCII coded)
2428 representation of the 16 bit SMB Tree Id field, which
2429 uniquely identifies the connection that submitted the job
2430 to the printer. The most significant digit of the numeric
2431 string shall be placed in octet position 2. All unused
2432 portions of this field shall be filled with spaces. The
2433 SMB Tree Id has a maximum value of 65,535.
2434 octets 41-48: The US-ASCII 8-decimal-digit leading zero
2435 representation of the File Handle returned from the device
2436 to the client in response to a Create Print File command.
2437 This format is reserved for agents.
2438
2439 'D' Transport Independent Printer/System Interface (TIP/SI)
2440 octets 2-40: Contains the Job Name from the Job Control-Start
2441 Job (JC-SJ) command. If the Job Name portion is less than
2442 40 octets, the left-most character in the string shall
2443 appear in octet position 2. Any unused portion of this
2444 field shall be filled with spaces. Otherwise, only the
2445 last 39 bytes shall be included.
2446 octets 41-48: The US-ASCII 8-decimal-digit leading zero
2447 representation of the jmJobIndex assigned by the agent.
2448 This format is reserved for agents, since the agent supplies
2449 octets 41-48, though the client supplies the job name. See
2450 format '1' reserved to clients to submit job name ids in
2451 which they supply octets 41-48.
2452
2453 'E' IPDS on the MVS or VSE platform
2454
2455 octets 2-40: Contains bytes 2-27 of the XOH Define Group
2456 Boundary Group ID triplet. Octet position 2 MUST carry the
2457 value x'01'. Bytes 28-40 MUST be filled with spaces.
2458 octets 41-48: The US-ASCII 8-decimal-digit leading zero
2459 representation of the jmJobIndex assigned by the agent.
2460 This format is reserved for agents, since the agent supplies
2461 octets 41-48, though the client supplies the job name.
2462

2463 'F' IPDS on the VM platform
2464 octets 2-40: Contains bytes 2-31 of the XOH Define Group
2465 Boundary Group ID triplet. Octet position 2 MUST carry the
2466 value x'02'. Bytes 32-40 MUST be filled with spaces.
2467 octets 41-48: The US-ASCII 8-decimal-digit leading zero
2468 representation of the jmJobIndex assigned by the agent.
2469 This format is reserved for agents, since the agent supplies
2470 octets 41-48, though the client supplies the file name.
2471
2472 'G' IPDS on the OS/400 platform
2473 octets 2-40: Contains bytes 2-36 of the XOH Define Group
2474 Boundary Group ID triplet. Octet position 2 MUST carry the
2475 value x'03'. Bytes 37-40 MUST be filled with spaces.
2476 octets 41-48: The US-ASCII 8-decimal-digit leading zero
2477 representation of the jmJobIndex assigned by the agent.
2478 This format is reserved for agents, since the agent supplies
2479 octets 41-48, though the client supplies the job name.
2480

2481 NOTE - the job submission id is only intended to be unique between a
2482 limited set of clients for a limited duration of time, namely, for the
2483 life time of the job in the context of the server or device that is
2484 processing the job. Some of the formats include something that is
2485 unique per client and a random number so that the same job submitted by
2486 the same client will have a different job submission id. For other
2487 formats, where part of the id is guaranteed to be unique for each
2488 client, such as the MAC address or URL, a sequential number SHOULD
2489 suffice for each client (and may be easier for each client to manage).
2490 Therefore, the length of the job submission id has been selected to
2491 reduce the probability of collision to an extremely low number, but is
2492 not intended to be an absolute guarantee of uniqueness. None-the-less,
2493 collisions are remotely possible, but without bad consequences, since
2494 this MIB is intended to be used only for monitoring jobs, not for
2495 controlling and managing them.

2496

2497

2498 3.6 Internationalization Considerations

2499 This section describes the internationalization considerations included
2500 in this MIB.

2501 3.6.1 Text generated by the server or device

2502 There are a few objects and attributes generated by the server or
2503 device that SHALL be represented using the Universal Multiple-Octet
2504 Coded Character Set (UCS) [ISO-10646]. These objects and attributes
2505 are always supplied (if implemented) by the agent, not by the job
2506 submitting client:

- 2507 1. jmGeneralJobSetName object
- 2508 2. processingMessage(6) attribute
- 2509 3. physicalDevice(32) (name value) attribute

2510 The character encoding scheme for representing these objects and
2511 attributes SHALL be UTF-8 as recommended by RFC 2130 [RFC 2130] and the
2512 "IETF Policy on Character Sets and Language" [char-set policy]. The
2513 'JmUTF8StringTC' textual convention is used to indicate UTF-8 text
2514 strings.

2515 NOTE - For strings in 7-bit US-ASCII, there is no impact since the UTF-
2516 8 representation of 7-bit ASCII is identical to the US-ASCII [US-ASCII]
2517 encoding.

2518 The text contained in the processingMessage(6) attribute is generated
2519 by the server/device. The natural language for the
2520 processingMessage(6) attribute is identified by the
2521 processingMessageNaturalLangTag(7) attribute. The
2522 processingMessageNaturalLangTag(7) attribute uses the
2523 JmNaturalLanguageTagTC textual convention which SHALL conform to the
2524 language tag mechanism specified in RFC 1766 [RFC-1766]. The
2525 JmNaturalLanguageTagTC value is the same as the IPP [IPP-model]
2526 'naturalLanguage' attribute syntax. RFC 1766 specifies that a US-ASCII
2527 string consisting of the natural language followed by an optional
2528 country field. Both fields use the same two-character codes from ISO
2529 639 [ISO-639] and ISO 3166 [ISO-3166], respectively, that are used in
2530 the Printer MIB for identifying language and country.

2531 Examples of the values of the processingMessageNaturalLangTag(7)
2532 attribute include:

- 2533 1. 'en' for English
- 2534 2. 'en-us' for US English
- 2535 3. 'fr' for French
- 2536 4. 'de' for German

2537 3.6.2 Text supplied by the job submitter

2538 All of the objects and attributes represented by the 'JmJobStringTC'
2539 textual-convention are either (1) supplied in the job submission
2540 protocol by the client that submits the job to the server or device or
2541 (2) are defaulted by the server or device if the job submitting client
2542 does not supply values. The agent SHALL represent these objects and
2543 attributes in the MIB either (1) in the coded character set as they
2544 were submitted or (2) MAY convert the coded character set to another
2545 coded character set or encoding scheme. In any case, the resulting
2546 coded character set representation SHOULD be UTF-8 [UTF-8], but SHALL
2547 be one in which the code positions from 0 to 31 is not used, 32 to 127
2548 is US-ASCII [US-ASCII], 127 is not unused, and the remaining code
2549 positions 128 to 255 represent single-byte or multi-byte graphic
2550 characters structured according to ISO 2022 [ISO 2022] or are unused.

2551 The coded character set SHALL be one of the ones registered with IANA
2552 [IANA] and SHALL be identified by the jobCodedCharSet attribute in the
2553 jmJobAttributeTable for the job. If the agent does not know what coded
2554 character set was used by the job submitting client, the agent SHALL
2555 either (1) return the 'unknown(2)' value for the jobCodedCharSet
2556 attribute or (2) not return the jobCodedCharSet attribute for the job.

2557 Examples of coded character sets which meet this criteria for use as
2558 the value of the jobCodedCharSet job attribute are: US-ASCII [US-
2559 ASCII], ISO 8859-1 (Latin-1) [ISO 8859-1], any ISO 8859-n, HP Roman8,
2560 IBM Code Page 850, Windows Default 8-bit set, UTF-8 [UTF-8], US-ASCII
2561 plus JIS X0208-1990 Japanese [JIS X0208], US-ASCII plus GB2312-1980 PRC
2562 Chinese [GB2312]. See the IANA registry of coded character sets [IANA
2563 charsets].

2564 Examples of coded character sets which do not meet this criteria are:
2565 national 7-bit sets conforming to ISO 646 (except US-ASCII), EBCDIC,
2566 and ISO 10646 (Unicode) [ISO-10646]. In order to represent Unicode
2567 characters, the UTF-8 [UTF-8] encoding scheme SHALL be used which has
2568 been assigned the MIBenum value of '106' by IANA.

2569 The jobCodedCharSet attribute uses the imported 'CodedCharSet' textual-
2570 convention from the Printer MIB [printmib].

2571 The natural language for attributes represented by the textual-
2572 convention JmJobStringTC is identified either (1) by the
2573 jobNaturalLanguageTag(9) attribute or is keywords in US-English (as in
2574 IPP). A monitoring application SHOULD attempt to localize keywords
2575 into the language of the user by means of some lookup mechanism. If
2576 the keyword value is not known to the monitoring application, the
2577 monitoring application SHOULD assume that the value is in the natural
2578 language specified by the job's jobNaturalLanguageTag(9) attribute and
2579 SHOULD present the value to its user as is. The
2580 jobNaturalLanguageTag(9) attribute value SHALL have the same syntax and
2581 semantics as the processingMessageNaturalLangTag(7) attribute, except
2582 that the jobNaturalLanguageTag(9) attribute identifies the natural
2583 language of attributes supplied by the job submitter instead of the
2584 natural language of the processingMessage(6) attribute. See Section
2585 3.6.1.

2586 3.6.3 'DateAndTime' for representing the date and time

2587 This MIB also contains objects that are represented using the
2588 DateAndTime textual convention from SMIV2 [SMIV2-TC]. The job
2589 management application SHALL display such objects in the locale of the
2590 user running the monitoring application.

2591 3.7 IANA and PWG Registration Considerations

2592 This MIB does not require any additional registration schemes for IANA,
2593 but does depend on registration schemes that other Internet standards

2594 track specifications have set up. The names of these IANA registration
2595 assignments under the /in-notes/iana/assignments/ path:

2596 1. printer-language-numbers - used as enums in the documentFormat(38)
2597 attribute

2598 2. media-types - uses as keywords in the documentFormat(38) attribute

2599 3. character-sets - used as enums in the jobCodedCharSet(8) attribute

2600 The Printer Working Group (PWG) will handle registration of additional
2601 enums after approving this standard, according to the procedures
2602 described in this section:

2603

2604 3.7.1 PWG Registration of enums

2605 This specification uses textual conventions to define enumerated values
2606 (enums) and bit values. Enumerations (enums) and bit values are sets
2607 of symbolic values defined for use with one or more objects or
2608 attributes. All enumeration sets and bit value sets are assigned a
2609 symbolic data type name (textual convention). As a convention the
2610 symbolic name ends in "TC" for textual convention. These enumerations
2611 are defined at the beginning of the MIB module specification.

2612 The PWG has defined several type of enumerations for use in the Job
2613 Monitoring MIB and the Printer MIB[print-mib]. These types differ in
2614 the method employed to control the addition of new enumerations.
2615 Throughout this document, references to "type n enum", where n can be
2616 1, 2 or 3 can be found in the various tables. The definitions of these
2617 types of enumerations are:

2618 3.7.1.1 Type 1 enumerations

2619 Type 1 enumeration: All the values are defined in the Job Monitoring
2620 MIB specification (RFC for the Job Monitoring MIB). Additional
2621 enumerated values require a new RFC.

2622 There are no type 1 enums in the current draft.

2623 3.7.1.2 Type 2 enumerations

2624 Type 2 enumeration: An initial set of values are defined in the Job
2625 Monitoring MIB specification. Additional enumerated values are
2626 registered with the PWG.

2627 The following type 2 enums are contained in the current draft :

2628 1. JmUTF8StringTC

- 2629 2. JmJobStringTC
- 2630 3. JmNaturalLanguageTagTC
- 2631 4. JmTimeStampTC
- 2632 5. JmFinishingTC [same enum values as IPP "finishing" attribute]
- 2633 6. JmPrintQualityTC [same enum values as IPP "print-quality"
- 2634 attribute]
- 2635 7. JmTonerEconomyTC
- 2636 8. JmMediumTypeTC
- 2637 9. JmJobSubmissionIDTypeTC
- 2638 10. JmJobCollationTypeTC
- 2639 11. JmJobStateTC [same enum values as IPP "job-state" attribute]
- 2640 12. JmAttributeTypeTC

2641 For those textual conventions that have the same enum values as the
2642 indicated IPP Job attribute are simultaneously registered by the PWG
2643 for use with IPP [ipp-model] and the Job Monitoring MIB.

2644 3.7.1.3 Type 3 enumeration

2645 Type 3 enumeration: An initial set of values are defined in the Job
2646 Monitoring MIB specification. Additional enumerated values are
2647 registered through the PWG without PWG review.

2648 There are no type 3 enums in the current draft.

2649 3.7.2 PWG Registration of type 2 bit values

2650 This draft contains the following type 2 bit value textual-conventions:

- 2651 1. JmJobServiceTypesTC
- 2652 2. JmJobStateReasons1TC
- 2653 3. JmJobStateReasons2TC
- 2654 4. JmJobStateReasons3TC
- 2655 5. JmJobStateReasons4TC

2656 These textual-conventions are defined as bits in an Integer so that
2657 they can be used with SNMPv1 SMI. The jobStateReasonsN (N=1..4)
2658 attributes are defined as bit values using the corresponding
2659 JmJobStateReasonsMTC textual-conventions.

2660 The registration of JmJobServiceTypesTC and JmJobStateReasonsMTC bit
2661 values follow the procedures for a type 2 enum as specified in Section
2662 3.7.1.2.

2663 3.7.3 PWG Registration of Job Submission Id Formats

2664 In addition to enums and bit values, this specification assigns a
2665 single ASCII digit or letter to various job submission ID formats. See
2666 the JmJobSubmissionIDTypeTC textual-convention and the object. The

2667 registration of JobSubmissionID format numbers follows the procedures
2668 for a type 2 enum as specified in Section 3.7.1.2.

2669 3.7.4 PWG Registration of MIME types/sub-types for document-formats

2670 The documentFormat(38) attribute has MIME type/sub-type values for
2671 indicating document formats which IANA registers as "media type" names.
2672 The values of the documentFormat(38) attribute are the same as the
2673 corresponding Internet Printing Protocol (IPP) "document-format" Job
2674 attribute values [ipp-model].

2675 3.8 Security Considerations

2676 3.8.1 Read-Write objects

2677 All objects are read-only, greatly simplifying the security
2678 considerations. If another MIB augments this MIB, that MIB might
2679 accept SNMP Write operations to objects in that MIB whose effect is to
2680 modify the values of read-only objects in this MIB. However, that MIB
2681 SHALL have to support the required access control in order to achieve
2682 security, not this MIB.

2683 3.8.2 Read-Only Objects In Other User's Jobs

2684 The security policy of some sites MAY be that unprivileged users can
2685 only get the objects from jobs that they submitted, plus a few minimal
2686 objects from other jobs, such as the jmJobKOctetsPerCopyRequested and
2687 jmJobKOctetsProcessed objects, so that a user can tell how busy a
2688 printer is. Other sites MAY allow all unprivileged users to see all
2689 objects of all jobs. This MIB does not require, nor does it specify
2690 how, such restrictions would be implemented. A monitoring application
2691 SHOULD enforce the site security policy with respect to returning
2692 information to an unprivileged end user that is using the monitoring
2693 application to monitor jobs that do not belong to that user, i.e., the
2694 jmJobOwner object in the jmJobTable does not match the user's user
2695 name.

2696 An operator is a privileged user that would be able to see all objects
2697 of all jobs, independent of the policy for unprivileged users.

2698 3.9 Notifications

2699 This MIB does not specify any notifications. For simplicity,
2700 management applications are expected to poll for status. The
2701 jmGeneralJobPersistence and jmGeneralAttributePersistence objects
2702 assist an application to determine the polling rate. The resulting
2703 network traffic is not expected to be significant.

2704 4 MIB specification

2705 The following pages constitute the actual Job Monitoring MIB.

```
2706 Job-Monitoring-MIB DEFINITIONS ::= BEGIN
2707
2708 IMPORTS
    MODULE-IDENTITY, OBJECT-TYPE, enterprises,
    Integer32                                FROM SNMPv2-SMI
    TEXTUAL-CONVENTION                       FROM SNMPv2-TC
    MODULE-COMPLIANCE, OBJECT-GROUP         FROM SNMPv2-CONF;
    -- The following textual-conventions are needed to implement
    -- certain attributes, but are not needed to compile this MIB.
    -- They are provided here for convenience:
    -- hrDeviceIndex                         FROM HOST-RESOURCES-MIB
    -- DateAndTime                           FROM SNMPv2-TC
    -- PrtInterpreterLangFamilyTC,
    -- CodedCharSet                          FROM Printer-MIB

2709
2710 -- Use the enterprises arc assigned to the PWG which is pwg(2699).
2711 -- Group all PWG mibs under mibs(1).
2712
2713 jobmonMIB MODULE-IDENTITY
2714     LAST-UPDATED "9811080000Z"
2715     ORGANIZATION "Printer Working Group (PWG)"
2716     CONTACT-INFO
2717         "Tom Hastings
2718         Postal:  Xerox Corp.
2719                 Mail stop ESAE-231
2720                 701 S. Aviation Blvd.
2721                 El Segundo, CA 90245
2722
2723         Tel:      (301)333-6413
2724         Fax:      (301)333-5514
2725         E-mail:   hastings@cpl0.es.xerox.com
2726
2727         Send questions and comments to the Printer Working Group (PWG)
2728         using the Job Monitoring Project (JMP) Mailing List:
2729         jmp@pwg.org
2730
2731         For further information, including how to subscribe to the
2732         jmp mailing list, access the PWG web page under 'JMP':
2733
2734         http://www.pwg.org/
2735
2736         Implementers of this specification are encouraged to join the
2737         jmp mailing list in order to participate in discussions on any
2738         clarifications needed and registration proposals being reviewed
2739         in order to achieve consensus."
2740     DESCRIPTION
2741         "The MIB module for monitoring job in servers, printers, and
2742         other devices.
2743
2744         Version: 1.3"
2745     ::= { enterprises pwg(2699) mibs(1) jobmonMIB(1) }
```

```
2746
2747 -- Textual conventions for this MIB module
2748
2749 JmUTF8StringTC ::= TEXTUAL-CONVENTION
2750     DISPLAY-HINT "255a"
2751     STATUS      current
2752     DESCRIPTION
2753         "To facilitate internationalization, this TC represents
2754         information taken from the ISO/IEC IS 10646-1 character set,
2755         encoded as an octet string using the UTF-8 character encoding
2756         scheme.
2757
2758         See section 3.6.1, entitled: 'Text generated by the server or
2759         device'."
2760     SYNTAX      OCTET STRING (SIZE (0..63))
2761
2762
2763
2764
2765 JmJobStringTC ::= TEXTUAL-CONVENTION
2766     STATUS      current
2767     DESCRIPTION
2768         "To facilitate internationalization, this TC represents
2769         information using any coded character set registered by IANA as
2770         specified in section 3.7. While it is recommended that the
2771         coded character set be UTF-8 [UTF-8], the actual coded
2772         character set SHALL be indicated by the value of the
2773         jobCodedCharSet(8) attribute for the job.
2774
2775         See section 3.6.2, entitled: 'Text supplied by the job
2776         submitter'."
2777     SYNTAX      OCTET STRING (SIZE (0..63))
2778
2779
2780
2781
2782 JmNaturalLanguageTagTC ::= TEXTUAL-CONVENTION
2783     STATUS      current
2784     DESCRIPTION
2785         "An IETF RFC 1766-compliant 'language tag', with zero or more
2786         sub-tags that identify a natural language. While RFC 1766
2787         specifies that the US-ASCII values are case-insensitive, this
2788         MIB specification requires that all characters SHALL be lower
2789         case in order to simplify comparing by management applications.
2790
2791         See section 3.6.1, entitled: 'Text generated by the server or
2792         device' and section 3.6.2, entitled: 'Text supplied by the job
2793         submitter'."
2794     SYNTAX      OCTET STRING (SIZE (0..63))
2795
2796
2797
2798 JmTimeStampTC ::= TEXTUAL-CONVENTION
```

```
2798     STATUS      current
2799     DESCRIPTION
2800         "The simple time at which an event took place.  The units are
2801         in seconds since the system was booted.
2802
2803         NOTE - JmTimeStampTC is defined in units of seconds, rather
2804         than 100ths of seconds, so as to be simpler for agents to
2805         implement (even if they have to implement the 100ths of a
2806         second to comply with implementing sysUpTime in MIB-II[mib-
2807         II].)
2808
2809         NOTE - JmTimeStampTC is defined as an Integer32 so that it can
2810         be used as a value of an attribute, i.e., as a value of the
2811         jmAttributeValueAsInteger object.  The TimeStamp textual-
2812         convention defined in SNMPv2-TC [SMIV2-TC] is defined as an
2813         APPLICATION 3 IMPLICIT INTEGER tag, not an Integer32 which is
2814         defined in SNMPv2-SMI [SMIV2-TC] as UNIVERSAL 2 IMPLICIT
2815         INTEGER, so cannot be used in this MIB as one of the values of
2816         jmAttributeValueAsInteger."
2817     SYNTAX      INTEGER (0..2147483647)
2818
2819
2820
2821
2822 JmJobSourcePlatformTypeTC ::= TEXTUAL-CONVENTION
2823     STATUS      current
2824     DESCRIPTION
2825         "The source platform type that can submit jobs to servers or
2826         devices in any of the 3 configurations.
2827
2828         This is a type 2 enumeration.  See Section 3.7.1.2.  See also
2829         IANA operating-system-names registry."
2830     SYNTAX      INTEGER {
2831         other(1),
2832         unknown(2),
2833         sptUNIX(3),           -- UNIX
2834         sptOS2(4),           -- OS/2
2835         sptPCDOS(5),         -- DOS
2836         sptNT(6),           -- NT
2837         sptMVS(7),          -- MVS
2838         sptVM(8),           -- VM
2839         sptOS400(9),        -- OS/400
2840         sptVMS(10),         -- VMS
2841         sptWindows(11),     -- Windows
2842         sptNetWare(12)      -- NetWare
2843     }
```



```
2833
2834 JmFinishingTC ::= TEXTUAL-CONVENTION
2835     STATUS      current
2836     DESCRIPTION
2837         "The type of finishing operation.
2838
2839         These values are the same as the enum values of the IPP
2840         'finishings' attribute.  See Section 3.7.1.2.
2841
2842         other(1),
2843             Some other finishing operation besides one of the specified
2844             or registered values.
2845
2846         unknown(2),
2847             The finishing is unknown.
2848
2849         none(3),
2850             Perform no finishing.
2851
2852         staple(4),
2853             Bind the document(s) with one or more staples. The exact
2854             number and placement of the staples is site-defined.
2855
2856         punch(5),
2857             This value indicates that holes are required in the
2858             finished document. The exact number and placement of the
2859             holes is site-defined. The punch specification MAY be
2860             satisfied (in a site- and implementation-specific manner)
2861             either by drilling/punching, or by substituting pre-drilled
2862             media.
2863
2864         cover(6),
2865             This value is specified when it is desired to select a non-
2866             printed (or pre-printed) cover for the document. This does
2867             not supplant the specification of a printed cover (on cover
2868             stock medium) by the document itself.
2869
2870         bind(7)
2871             This value indicates that a binding is to be applied to the
2872             document; the type and placement of the binding is product-
2873             specific.
2874
2875         This is a type 2 enumeration.  See Section 3.7.1.2."
2876     SYNTAX      INTEGER {
2877         other(1),
2878         unknown(2),
2879         none(3),
2880         staple(4),
2881         punch(5),
2882         cover(6),
2883         bind(7)
2884     }
```

```
2885
2886
2887 JmPrintQualityTC ::= TEXTUAL-CONVENTION
2888     STATUS      current
2889     DESCRIPTION
2890         "Print quality settings.
2891
2892         These values are the same as the enum values of the IPP 'print-
2893         quality' attribute.  See Section 3.7.1.2.
2894
2895         This is a type 2 enumeration.  See Section 3.7.1.2."
2896     SYNTAX      INTEGER {
2897         other(1),      -- Not one of the specified or registered
2898                       -- values.
2899         unknown(2),   -- The actual value is unknown.
2900         draft(3),     -- Lowest quality available on the printer.
2901         normal(4),    -- Normal or intermediate quality on the
2902                       -- printer.
2903         high(5)       -- Highest quality available on the printer.
2904     }
2905
2906
2907
2908
2909
2910
2911
2912
2913
2914
2915
2916
2917
2918
2919
2920
2921
2922
2923 JmPrinterResolutionTC ::= TEXTUAL-CONVENTION
2924     STATUS      current
2925     DESCRIPTION
2926         "Printer resolutions.
2927
2928         Nine octets consisting of two 4-octet SIGNED-INTEGERS followed
2929         by a SIGNED-BYTE.  The values are the same as those specified
2930         in the Printer MIB [printmib].  The first SIGNED-INTEGER
2931         contains the value of prtMarkerAddressabilityXFeedDir.  The
2932         second SIGNED-INTEGER contains the value of
2933         prtMarkerAddressabilityFeedDir.  The SIGNED-BYTE contains the
2934         value of prtMarkerAddressabilityUnit.
2935
2936         Note: the latter value is either 3 (tenThousandsOfInches) or 4
2937         (micrometers) and the addressability is in 10,000 units of
2938         measure.  Thus the SIGNED-INTEGERS represent integral values in
2939         either dots-per-inch or dots-per-centimeter.
2940
2941         The syntax is the same as the IPP 'printer-resolution'
2942         attribute.  See Section 3.7.1.2."
2943     SYNTAX      OCTET STRING (SIZE(9))
```

```
2924
2925 JmTonerEconomyTC ::= TEXTUAL-CONVENTION
2926     STATUS      current
2927     DESCRIPTION
2928         "Toner economy settings.
2929
2930         This is a type 2 enumeration.  See Section 3.7.1.2."
2931     SYNTAX      INTEGER {
2932         unknown(2),      -- unknown.
2933         off(3),          -- Off. Normal. Use full toner.
2934         on(4)            -- On. Use less toner than normal.
2935     }
2936
2937 JmBooleanTC ::= TEXTUAL-CONVENTION
2938     STATUS      current
2939     DESCRIPTION
2940         "Boolean true or false value.
2941
2942         This is a type 2 enumeration.  See Section 3.7.1.2."
2943     SYNTAX      INTEGER {
2944         unknown(2),      -- unknown.
2945         false(3),        -- FALSE.
2946         true(4)          -- TRUE.
2947     }
2948
2949 JmMediumTypeTC ::= TEXTUAL-CONVENTION
2950     STATUS      current
2951     DESCRIPTION
2952         "Identifies the type of medium.
2953
2954         other(1),
2955             The type is neither one of the values listed in this
2956             specification nor a registered value.
2957
2958         unknown(2),
2959             The type is not known.
2960
2961         stationery(3),
2962             Separately cut sheets of an opaque material.
2963
2964         transparency(4),
2965             Separately cut sheets of a transparent material.
2966
2967         envelope(5),
2968             Envelopes that can be used for conventional mailing
2969             purposes.
```

2968
2969 envelopePlain(6),
2970 Envelopes that are not preprinted and have no windows.
2971
2972 envelopeWindow(7),
2973 Envelopes that have windows for addressing purposes.
2974
2975 continuousLong(8),
2976 Continuously connected sheets of an opaque material
2977 connected along the long edge.
2978
2979 continuousShort(9),
2980 Continuously connected sheets of an opaque material
2981 connected along the short edge.
2982
2983 tabStock(10),
2984 Media with tabs.
2985
2986 multiPartForm(11),
2987 Form medium composed of multiple layers not pre-attached to
2988 one another; each sheet MAY be drawn separately from an
2989 input source.
2990
2991 labels(12),
2992 Label-stock.
2993
2994 multiLayer(13)
2995 Form medium composed of multiple layers which are pre-
2996 attached to one another, e.g. for use with impact printers.
2997
2998 This is a type 2 enumeration. See Section 3.7.1.2. These enum
2999 values correspond to the keyword name strings of the
3000 prtInputMediaType object in the Printer MIB [print-mib]. There
3001 is no printer description attribute in IPP/1.0 that represents
3002 these values."
3003 SYNTAX INTEGER {
3004 other(1),
3005 unknown(2),
3006 stationery(3),
3007 transparency(4),
3008 envelope(5),
3009 envelopePlain(6),
3010 envelopeWindow(7),
3011 continuousLong(8),
3012 continuousShort(9),
3013 tabStock(10),
3014 multiPartForm(11),
3015 labels(12),
3016 multiLayer(13)
3017 }
3018
3019

```
3020 JmJobCollationTypeTC ::= TEXTUAL-CONVENTION
3021     STATUS      current
3022     DESCRIPTION
3023         "This value is the type of job collation. Implementations that
3024         don't support multiple documents or don't support multiple
3025         copies SHALL NOT support the uncollatedDocuments(5) value.
3026
3027         This is a type 2 enumeration. See Section 3.7.1.2. See also
3028         Section 3.4, entitled 'Monitoring Job Progress'."
3029     SYNTAX      INTEGER {
3030         other(1),
3031         unknown(2),
3032         uncollatedSheets(3),      -- sheets within each document copy
3033                                   -- are not collated: 1 1 ..., 2 2 ...,
3034                                   -- No corresponding value of IPP
3035                                   -- "multiple-document-handling"
3036         collatedDocuments(4),    -- internal collated sheets,
3037                                   -- documents: A, B, A, B, ...
3038                                   -- Corresponds to IPP "multiple-
3039                                   -- document-handling"='separate-
3040                                   -- documents-collated-copies'
3041         uncollatedDocuments(5)  -- internal collated sheets,
3042                                   -- documents: A, A, ..., B, B, ...
3043                                   -- Corresponds to IPP "multiple-
3044                                   -- document-handling"='separate-
3045                                   -- documents-uncollated-copies'
3046     }
3047
3048
3049 JmJobSubmissionIDTypeTC ::= TEXTUAL-CONVENTION
3050     STATUS      current
3051     DESCRIPTION
3052         "Identifies the format type of a job submission ID.
3053
3054         Each job submission ID is a fixed-length, 48-octet printable
3055         US-ASCII [US-ASCII] coded character string containing no
3056         control characters, consisting of the fields defined in section
3057         3.5.1.
3058
3059         This is like a type 2 enumeration. See section 3.7.3."
3060     SYNTAX      OCTET STRING(SIZE(1)) -- ASCII '0'-'9', 'A'-'Z', 'a'-'z'
```

```

3061
3062 JmJobStateTC ::= TEXTUAL-CONVENTION
3063     STATUS      current
3064     DESCRIPTION
3065         "The current state of the job (pending, processing, completed,
3066         etc.).
3067
3068     The following figure shows the normal job state transitions:
3069
3070
3071
3072     +----> pending(3)  -----> processing(5)  -----+-----> canceled(7)
3073     |                ^                ^                /
3074     --->+            |                |                \
3075     |                v                v                /
3076     +----> pendingHeld(4)  processingStopped(6)  ----+-----> aborted(8)
3077

```

Figure 4 - Normal Job State Transitions

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

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

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

unknown(2),

The job state is *not* known, or its state is indeterminate.

pending(3),

The job is a candidate to start processing, but is not yet processing.

pendingHeld(4),

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

3112 JmJobStateReasonsMTC (N=1..4) textual convention for the
3113 specification of each reason.
3114
3115 processing(5),
3116 One or more of:
3117
3118 1. the job is using, or is attempting to use, one or more
3119 purely software processes that are analyzing, creating, or
3120 interpreting a PDL, etc.,
3121
3122 2. the job is using, or is attempting to use, one or more
3123 hardware devices that are interpreting a PDL, making marks
3124 on a medium, and/or performing finishing, such as stapling,
3125 etc.,
3126
3127 OR
3128
3129 3. (configuration 2) the server has made the job ready for
3130 printing, but the output device is not yet printing it,
3131 either because the job hasn't reached the output device or
3132 because the job is queued in the output device or some
3133 other spooler, awaiting the output device to print it.
3134
3135 When the job is in the processing state, the entire job
3136 state includes the detailed status represented in the
3137 device MIB indicated by the hrDeviceIndex value of the
3138 job's physicalDevice attribute, if the agent implements
3139 such a device MIB.
3140
3141 Implementations MAY, though they NEED NOT, include
3142 additional values in the job's jmJobStateReasons1 object to
3143 indicate the progress of the job, such as adding the
3144 jobPrinting value to indicate when the device is actually
3145 making marks on a medium and/or the processingToStopPoint
3146 value to indicate that the server or device is in the
3147 process of canceling or aborting the job.
3148
3149 processingStopped(6),
3150 The job has stopped while processing for any number of
3151 reasons and will return to the processing state as soon as
3152 the reasons are no longer present.
3153
3154 The job's jmJobStateReasons1 object and/or the job's
3155 jobStateReasonsN (N=2..4) attributes MAY indicate why the
3156 job has stopped processing. For example, if the output
3157 device is stopped, the deviceStopped value MAY be included
3158 in the job's jmJobStateReasons1 object.
3159
3160 NOTE - When an output device is stopped, the device usually
3161 indicates its condition in human readable form at the
3162 device. The management application can obtain more
3163 complete device status remotely by querying the appropriate

3164 device MIB using the job's deviceIndex attribute(s), if the
3165 agent implements such a device MIB
3166
3167 canceled(7),
3168 A client has canceled the job and the server or device has
3169 completed canceling the job AND all MIB objects and
3170 attributes have reached their final values for the job.
3171 While the server or device is canceling the job, the job's
3172 jmJobStateReasons1 object SHOULD contain the
3173 processingToStopPoint value and one of the canceledByUser,
3174 canceledByOperator, or canceledAtDevice values. The
3175 canceledByUser, canceledByOperator, or canceledAtDevice
3176 values remain while the job is in the canceled state.
3177
3178 aborted(8),
3179 The job has been aborted by the system, usually while the
3180 job was in the processing or processingStopped state and
3181 the server or device has completed aborting the job AND all
3182 MIB objects and attributes have reached their final values
3183 for the job. While the server or device is aborting the
3184 job, the job's jmJobStateReasons1 object MAY contain the
3185 processingToStopPoint and abortedBySystem values. If
3186 implemented, the abortedBySystem value SHALL remain while
3187 the job is in the aborted state.
3188
3189 completed(9)
3190 The job has completed successfully or with warnings or
3191 errors after processing and all of the media have been
3192 successfully stacked in the appropriate output bin(s) AND
3193 all MIB objects and attributes have reached their final
3194 values for the job. The job's jmJobStateReasons1 object
3195 SHOULD contain one of: completedSuccessfully,
3196 completedWithWarnings, or completedWithErrors values.
3197
3198 This is a type 2 enumeration. See Section 3.7.1.2."
3199 SYNTAX INTEGER {
3200 unknown(2),
3201 pending(3),
3202 pendingHeld(4),
3203 processing(5),
3204 processingStopped(6),
3205 canceled(7),
3206 aborted(8),
3207 completed(9)
3208 }


```
3209
3210 JmAttributeTypeTC ::= TEXTUAL-CONVENTION
3211     STATUS      current
3212     DESCRIPTION
3213         "The type of the attribute which identifies the attribute.
3214
3215     NOTE - The enum assignments are grouped logically with values
3216     assigned in groups of 20, so that additional values may be
3217     registered in the future and assigned a value that is part of
3218     their logical grouping.
3219
3220     Values in the range 2**30 to 2**31-1 are reserved for private
3221     or experimental usage. This range corresponds to the same
3222     range reserved in IPP. Implementers are warned that use of
3223     such values may conflict with other implementations.
3224     Implementers are encouraged to request registration of enum
3225     values following the procedures in Section 3.7.1.
3226
3227     See Section 3.2 entitled 'The Attribute Mechanism' for a
3228     description of this textual-convention and its use in the
3229     jmAttributeTable. See Section 3.3.8 for the specification of
3230     each attribute. The comment(s) after each enum assignment
3231     specifies the data type(s) of the attribute.
3232
3233     This is a type 2 enumeration. See Section 3.7.1.2."
3234
3235     SYNTAX      INTEGER {
3236         other(1),                -- Integer32 (-2..2147483647)
3237                                 -- AND/OR
3238                                 -- OCTET STRING(SIZE(0..63))
3239
3240         -- Job State attributes:
3241         jobStateReasons2(3),     -- JmJobStateReasons2TC
3242         jobStateReasons3(4),     -- JmJobStateReasons3TC
3243         jobStateReasons4(5),     -- JmJobStateReasons4TC
3244         processingMessage(6),    -- JmUTF8StringTC (SIZE(0..63))
3245         processingMessageNaturalLangTag(7),
3246                                 -- OCTET STRING(SIZE(0..63))
3247         jobCodedCharSet(8),      -- CodedCharSet
3248         jobNaturalLanguageTag(9), -- OCTET STRING(SIZE(0..63))
3249
```

```
3250      -- Job Identification attributes:
3251      jobURI(20),                -- OCTET STRING(SIZE(0..63))
3252      jobAccountName(21),        -- OCTET STRING(SIZE(0..63))
3253      serverAssignedJobName(22), -- JmJobStringTC (SIZE(0..63))
3254      jobName(23),               -- JmJobStringTC (SIZE(0..63))
3255      jobServiceTypes(24),       -- JmJobServiceTypesTC
3256      jobSourceChannelIndex(25), -- Integer32 (0..2147483647)
3257      jobSourcePlatformType(26), -- JmJobSourcePlatformTypeTC
3258      submittingServerName(27),  -- JmJobStringTC (SIZE(0..63))
3259      submittingApplicationName(28), -- JmJobStringTC (SIZE(0..63))
3260      jobOriginatingHost(29),    -- JmJobStringTC (SIZE(0..63))
3261      deviceNameRequested(30),   -- JmJobStringTC (SIZE(0..63))
3262      queueNameRequested(31),   -- JmJobStringTC (SIZE(0..63))
3263      physicalDevice(32),        -- hrDeviceIndex
3264                                  -- AND/OR
3265                                  -- JmUTF8StringTC (SIZE(0..63))
3266      numberOfDocuments(33),     -- Integer32 (-2..2147483647)
3267      fileName(34),              -- JmJobStringTC (SIZE(0..63))
3268      documentName(35),          -- JmJobStringTC (SIZE(0..63))
3269      jobComment(36),            -- JmJobStringTC (SIZE(0..63))
3270      documentFormatIndex(37),   -- Integer32 (0..2147483647)
3271      documentFormat(38),        -- PrtInterpreterLangFamilyTC
3272                                  -- AND/OR
3273                                  -- OCTET STRING(SIZE(0..63))
3274
3275      -- Job Parameter attributes:
3276      jobPriority(50),             -- Integer32 (-2..100)
3277      jobProcessAfterDateAndTime(51), -- DateAndTime (SNMPv2-TC)
3278      jobHold(52),                -- JmBooleanTC
3279      jobHoldUntil(53),           -- JmJobStringTC (SIZE(0..63))
3280      outputBin(54),              -- Integer32 (0..2147483647)
3281                                  -- AND/OR
3282                                  -- JmJobStringTC (SIZE(0..63))
3283      sides(55),                  -- Integer32 (-2..2)
3284      finishing(56),              -- JmFinishingTC
3285
3286      -- Image Quality attributes:
3287      printQualityRequested(70),   -- JmPrintQualityTC
3288      printQualityUsed(71),        -- JmPrintQualityTC
3289      printerResolutionRequested(72), -- JmPrinterResolutionTC
3290      printerResolutionUsed(73),    -- JmPrinterResolutionTC
3291      tonerEcomonyRequested(74),    -- JmTonerEconomyTC
3292      tonerEcomonyUsed(75),         -- JmTonerEconomyTC
3293      tonerDensityRequested(76),    -- Integer32 (-2..100)
3294      tonerDensityUsed(77),         -- Integer32 (-2..100)
3295
```

```

3296      -- Job Progress attributes:
3297      jobCopiesRequested(90),          -- Integer32 (-2..2147483647)
3298      jobCopiesCompleted(91),         -- Integer32 (-2..2147483647)
3299      documentCopiesRequested(92),    -- Integer32 (-2..2147483647)
3300      documentCopiesCompleted(93),    -- Integer32 (-2..2147483647)
3301      jobKOctetsTransferred(94),     -- Integer32 (-2..2147483647)
3302      sheetCompletedCopyNumber(95),   -- Integer32 (-2..2147483647)
3303      sheetCompletedDocumentNumber(96),
3304                                          -- Integer32 (-2..2147483647)
3305      jobCollationType(97),          -- JmJobCollationTypeTC
3306
3307      -- Impression attributes:
3308      impressionsSpooled(110),         -- Integer32 (-2..2147483647)
3309      impressionsSentToDevice(111),   -- Integer32 (-2..2147483647)
3310      impressionsInterpreted(112),    -- Integer32 (-2..2147483647)
3311      impressionsCompletedCurrentCopy(113),
3312                                          -- Integer32 (-2..2147483647)
3313      fullColorImpressionsCompleted(114),
3314                                          -- Integer32 (-2..2147483647)
3315      highlightColorImpressionsCompleted(115),
3316                                          -- Integer32 (-2..2147483647)
3317
3318      -- Page attributes:
3319      pagesRequested(130),            -- Integer32 (-2..2147483647)
3320      pagesCompleted(131),           -- Integer32 (-2..2147483647)
3321      pagesCompletedCurrentCopy(132), -- Integer32 (-2..2147483647)
3322
3323      -- Sheet attributes:
3324      sheetsRequested(150),           -- Integer32 (-2..2147483647)
3325      sheetsCompleted(151),          -- Integer32 (-2..2147483647)
3326      sheetsCompletedCurrentCopy(152), -- Integer32 (-2..2147483647)
3327
3328      -- Resource attributes:
3329      mediumRequested(170),           -- JmMediumTypeTC
3330                                          -- AND/OR
3331                                          -- JmJobStringTC (SIZE(0..63))
3332      mediumConsumed(171),           -- Integer32 (-2..2147483647)
3333                                          -- AND
3334                                          -- JmJobStringTC (SIZE(0..63))
3335      colorantRequested(172),        -- Integer32 (-2..2147483647)
3336                                          -- AND/OR
3337                                          -- JmJobStringTC (SIZE(0..63))
3338      colorantConsumed(173),         -- Integer32 (-2..2147483647)
3339                                          -- AND/OR
3340                                          -- JmJobStringTC (SIZE(0..63))
3341      mediumTypeConsumed(174),        -- Integer32 (-2..2147483647)
3342                                          -- AND
3343                                          -- JmJobStringTC (SIZE(0..63))
3344      mediumSizeConsumed(175),        -- Integer32 (-2..2147483647)
3345                                          -- AND
3346                                          -- JmJobStringTC (SIZE(0..63))
3347

```

```
3348      -- Time attributes:
3349      jobSubmissionToServerTime(190), -- JmTimeStampTC
3350                                         -- AND/OR
3351                                         -- DateAndTime
3352      jobSubmissionTime(191),          -- JmTimeStampTC
3353                                         -- AND/OR
3354                                         -- DateAndTime
3355      jobStartedBeingHeldTime(192),    -- JmTimeStampTC
3356                                         -- AND/OR
3357                                         -- DateAndTime
3358      jobStartedProcessingTime(193),   -- JmTimeStampTC
3359                                         -- AND/OR
3360                                         -- DateAndTime
3361      jobCompletionTime(194),          -- JmTimeStampTC
3362                                         -- AND/OR
3363                                         -- DateAndTime
3364      jobProcessingCPUTime(195)        -- Integer32 (-2..2147483647)
3365  }
3366
```

3367 JmJobServiceTypesTC ::= TEXTUAL-CONVENTION
3368 STATUS current
3369 DESCRIPTION
3370 "Specifies the type(s) of service to which the job has been
3371 submitted (print, fax, scan, etc.). The service type is
3372 represented as an enum that is bit encoded with each job
3373 service type so that more general and arbitrary services can be
3374 created, such as services with more than one destination type,
3375 or ones with only a source or only a destination. For example,
3376 a job service might scan, faxOut, and print a single job. In
3377 this case, three bits would be set in the jobServiceTypes
3378 attribute, corresponding to the hexadecimal values: 0x8 + 0x20
3379 + 0x4, respectively, yielding: 0x2C.
3380
3381 Whether this attribute is set from a job attribute supplied by
3382 the job submission client or is set by the recipient job
3383 submission server or device depends on the job submission
3384 protocol. With either implementation, the agent SHALL return a
3385 non-zero value for this attribute indicating the type of the
3386 job.
3387
3388 One of the purposes of this attribute is to permit a requester
3389 to filter out jobs that are not of interest. For example, a
3390 printer operator MAY only be interested in jobs that include
3391 printing. That is why the attribute is in the job
3392 identification category.
3393
3394 The following service component types are defined (in
3395 hexadecimal) and are assigned a separate bit value for use with
3396 the jobServiceTypes attribute:
3397
3398 other 0x1
3399 The job contains some instructions that are not one of the
3400 identified types.
3401
3402 unknown 0x2
3403 The job contains some instructions whose type is unknown to
3404 the agent.
3405
3406 print 0x4
3407 The job contains some instructions that specify printing
3408
3409 scan 0x8
3410 The job contains some instructions that specify scanning
3411
3412 faxIn 0x10
3413 The job contains some instructions that specify receive fax
3414
3415 faxOut 0x20
3416 The job contains some instructions that specify sending fax
3417

```
3418         getFile                0x40
3419             The job contains some instructions that specify accessing
3420             files or documents
3421
3422         putFile                  0x80
3423             The job contains some instructions that specify storing
3424             files or documents
3425
3426         mailList                 0x100
3427             The job contains some instructions that specify
3428             distribution of documents using an electronic mail system.
3429
3430             These bit definitions are the equivalent of a type 2 enum
3431             except that combinations of them MAY be used together.  See
3432             section 3.7.1.2."
3433     SYNTAX          INTEGER (0..2147483647)    -- 31 bits, all but sign bit
3434
3435
3436
3437 JmJobStateReasons1TC ::= TEXTUAL-CONVENTION
3438     STATUS          current
3439     DESCRIPTION
3440         "The JmJobStateReasonsMTC (N=1..4) textual-conventions are used
3441         with the jmJobStateReasons1 object and jobStateReasonsN
3442         (N=2..4), respectively, to provide additional information
3443         regarding the current jmJobState object value.  These values
3444         MAY be used with any job state or states for which the reason
3445         makes sense.  See section 3.3.9.1 for the specification of each
3446         bit value defined for use with the JmJobStateReasons1TC.
3447
3448         These bit definitions are the equivalent of a type 2 enum
3449         except that combinations of bits may be used together.  See
3450         section 3.7.1.2."
3451     SYNTAX          INTEGER (0..2147483647)    -- 31 bits, all but sign bit
3452
3453
3454
3455 JmJobStateReasons2TC ::= TEXTUAL-CONVENTION
3456     STATUS          current
3457     DESCRIPTION
3458         "This textual-convention is used with the jobStateReasons2
3459         attribute to provides additional information regarding the
3460         jmJobState object.  See section 3.3.9.2 for the specification
3461         of JmJobStateReasons2TC.  See section 3.3.9.1 for the
3462         description under JmJobStateReasons1TC for additional
3463         information that applies to all reasons.
3464
3465         These bit definitions are the equivalent of a type 2 enum
3466         except that combinations of them may be used together.  See
3467         section 3.7.1.2."
3468     SYNTAX          INTEGER (0..2147483647)    -- 31 bits, all but sign bit
3469
```

```
3470 JmJobStateReasons3TC ::= TEXTUAL-CONVENTION
3471     STATUS      current
3472     DESCRIPTION
3473         "This textual-convention is used with the jobStateReasons3
3474         attribute to provides additional information regarding the
3475         jmJobState object.  See section 3.3.9.3 for the specification
3476         of JmJobStateReasons3TC.  See section 3.3.9.1 for the
3477         description under JmJobStateReasons1TC for additional
3478         information that applies to all reasons.
3479
3480         These bit definitions are the equivalent of a type 2 enum
3481         except that combinations of them may be used together.  See
3482         section 3.7.1.2.  The remaining bits are reserved for future
3483         standardization and/or registration."
3484     SYNTAX      INTEGER (0..2147483647)  -- 31 bits, all but sign bit
3485
3486
3487
3488
3489
3490 JmJobStateReasons4TC ::= TEXTUAL-CONVENTION
3491     STATUS      current
3492     DESCRIPTION
3493         "This textual-convention is used in the jobStateReasons4
3494         attribute to provides additional information regarding the
3495         jmJobState object.  See section 3.3.9.4 for the specification
3496         of JmJobStateReasons4TC.  See section 3.3.9.1 for the
3497         description under JmJobStateReasons1TC for additional
3498         information that applies to all reasons.
3499
3500         These bit definitions are the equivalent of a type 2 enum
3501         except that combinations of them may be used together.  See
3502         section 3.7.1.2."
3503     SYNTAX      INTEGER (0..2147483647)  -- 31 bits, all but sign bit
```

```

3504
3505 jobmonMIBObjects OBJECT IDENTIFIER ::= { jobmonMIB 1 }
3506
3507 -- The General Group (MANDATORY)
3508
3509 -- The jmGeneralGroup consists entirely of the jmGeneralTable.
3510
3511 jmGeneral OBJECT IDENTIFIER ::= { jobmonMIBObjects 1 }
3512
3513 jmGeneralTable OBJECT-TYPE
3514     SYNTAX      SEQUENCE OF JmGeneralEntry
3515     MAX-ACCESS  not-accessible
3516     STATUS      current
3517     DESCRIPTION
3518         "The jmGeneralTable consists of information of a general nature
3519         that are per-job-set, but are not per-job. See Section 2
3520         entitled 'Terminology and Job Model' for the definition of a
3521         job set.
3522
3523         The MANDATORY-GROUP macro specifies that this group is
3524         MANDATORY."
3525     ::= { jmGeneral 1 }
3526
3527
3528 jmGeneralEntry OBJECT-TYPE
3529     SYNTAX      JmGeneralEntry
3530     MAX-ACCESS  not-accessible
3531     STATUS      current
3532     DESCRIPTION
3533         "Information about a job set (queue).
3534
3535         An entry SHALL exist in this table for each job set."
3536     INDEX      { jmGeneralJobSetIndex }
3537     ::= { jmGeneralTable 1 }
3538
3539
3540 JmGeneralEntry ::= SEQUENCE {
3541     jmGeneralJobSetIndex      Integer32 (1..32767),
3542     jmGeneralNumberOfActiveJobs Integer32 (0..2147483647),
3543     jmGeneralOldestActiveJobIndex Integer32 (0..2147483647),
3544     jmGeneralNewestActiveJobIndex Integer32 (0..2147483647),
3545     jmGeneralJobPersistence    Integer32 (15..2147483647),
3546     jmGeneralAttributePersistence Integer32 (15..2147483647),
3547     jmGeneralJobSetName        JmUTF8StringTC (SIZE(0..63))
3548 }
3549

```



```
3550 jmGeneralJobSetIndex OBJECT-TYPE
3551     SYNTAX      Integer32 (1..32767)
3552     MAX-ACCESS  not-accessible
3553     STATUS      current
3554     DESCRIPTION
3555         "A unique value for each job set in this MIB.  The jmJobTable
3556         and jmAttributeTable tables have this same index as their
3557         primary index.
3558
3559         The value(s) of the jmGeneralJobSetIndex SHALL be persistent
3560         across power cycles, so that clients that have retained
3561         jmGeneralJobSetIndex values will access the same job sets upon
3562         subsequent power-up.
3563
3564         An implementation that has only one job set, such as a printer
3565         with a single queue, SHALL hard code this object with the value
3566         1.
3567
3568         See Section 2 entitled 'Terminology and Job Model' for the
3569         definition of a job set.
3570         Corresponds to the first index in jmJobTable and
3571         jmAttributeTable."
3572     ::= { jmGeneralEntry 1 }
3573
3574
3575 jmGeneralNumberOfActiveJobs OBJECT-TYPE
3576     SYNTAX      Integer32 (0..2147483647)
3577     MAX-ACCESS  read-only
3578     STATUS      current
3579     DESCRIPTION
3580         "The current number of 'active' jobs in the jmJobIDTable,
3581         jmJobTable, and jmAttributeTable, i.e., the total number of
3582         jobs that are in the pending, processing, or processingStopped
3583         states.  See the JmJobStateTC textual-convention for the exact
3584         specification of the semantics of the job states."
3585     DEFVAL      { 0 }      -- no jobs
3586     ::= { jmGeneralEntry 2 }
3587
```

```
3588 jmGeneralOldestActiveJobIndex OBJECT-TYPE
3589     SYNTAX      Integer32 (0..2147483647)
3590     MAX-ACCESS  read-only
3591     STATUS      current
3592     DESCRIPTION
3593         "The jmJobIndex of the oldest job that is still in one of the
3594         'active' states (pending, processing, or processingStopped).
3595         In other words, the index of the 'active' job that has been in
3596         the job tables the longest.
3597
3598         If there are no active jobs, the agent SHALL set the value of
3599         this object to 0.
3600
3601         See Section 3.2 entitled 'The Job Tables and the Oldest Active
3602         and Newest Active Indexes' for a description of the usage of
3603         this object."
3604     DEFVAL      { 0 }      -- no active jobs
3605     ::= { jmGeneralEntry 3 }
3606
3607
3608
3609 jmGeneralNewestActiveJobIndex OBJECT-TYPE
3610     SYNTAX      Integer32 (0..2147483647)
3611     MAX-ACCESS  read-only
3612     STATUS      current
3613     DESCRIPTION
3614         "The jmJobIndex of the newest job that is in one of the
3615         'active' states (pending, processing, or processingStopped).
3616         In other words, the index of the 'active' job that has been
3617         most recently added to the job tables.
3618
3619         When all jobs become 'inactive', i.e., enter the pendingHeld,
3620         completed, canceled, or aborted states, the agent SHALL set the
3621         value of this object to 0.
3622
3623         See Section 3.2 entitled 'The Job Tables and the Oldest Active
3624         and Newest Active Indexes' for a description of the usage of
3625         this object."
3626     DEFVAL      { 0 }      -- no active jobs
3627     ::= { jmGeneralEntry 4 }
3628
```

```
3629 jmGeneralJobPersistence OBJECT-TYPE
3630     SYNTAX      Integer32 (15..2147483647)
3631     UNITS       "seconds"
3632     MAX-ACCESS  read-only
3633     STATUS      current
3634     DESCRIPTION
3635         "The minimum time in seconds for this instance of the Job Set
3636         that an entry SHALL remain in the jmJobIDTable and jmJobTable
3637         after processing has completed, i.e., the minimum time in
3638         seconds starting when the job enters the completed, canceled,
3639         or aborted state.
3640
3641         Configuring this object is implementation-dependent.
3642
3643         This value SHALL be equal to or greater than the value of
3644         jmGeneralAttributePersistence. This value SHOULD be at least
3645         60 which gives a monitoring or accounting application one
3646         minute in which to poll for job data."
3647     DEFVAL      { 60 }          -- one minute
3648     ::= { jmGeneralEntry 5 }
3649
3650
3651
3652 jmGeneralAttributePersistence OBJECT-TYPE
3653     SYNTAX      Integer32 (15..2147483647)
3654     UNITS       "seconds"
3655     MAX-ACCESS  read-only
3656     STATUS      current
3657     DESCRIPTION
3658         "The minimum time in seconds for this instance of the Job Set
3659         that an entry SHALL remain in the jmAttributeTable after
3660         processing has completed , i.e., the time in seconds starting
3661         when the job enters the completed, canceled, or aborted state.
3662
3663         Configuring this object is implementation-dependent.
3664
3665         This value SHOULD be at least 60 which gives a monitoring or
3666         accounting application one minute in which to poll for job
3667         data."
3668     DEFVAL      { 60 }          -- one minute
3669     ::= { jmGeneralEntry 6 }
3670
```

```
3671 jmGeneralJobSetName OBJECT-TYPE
3672     SYNTAX      JmUTF8StringTC (SIZE(0..63))
3673     MAX-ACCESS  read-only
3674     STATUS      current
3675     DESCRIPTION
3676         "The human readable name of this job set assigned by the system
3677         administrator (by means outside of this MIB).  Typically, this
3678         name SHOULD be the name of the job queue.  If a server or
3679         device has only a single job set, this object can be the
3680         administratively assigned name of the server or device itself.
3681         This name does not need to be unique, though each job set in a
3682         single Job Monitoring MIB SHOULD have distinct names.
3683
3684         NOTE - If the job set corresponds to a single printer and the
3685         Printer MIB is implemented, this value SHOULD be the same as
3686         the prtGeneralPrinterName object in the draft Printer MIB
3687         [print-mib-draft].  If the job set corresponds to an IPP
3688         Printer, this value SHOULD be the same as the IPP 'printer-
3689         name' Printer attribute.
3690
3691         NOTE - The purpose of this object is to help the user of the
3692         job monitoring application distinguish between several job sets
3693         in implementations that support more than one job set.
3694
3695         See the OBJECT compliance macro for the minimum maximum length
3696         required for conformance."
3697     DEFVAL      { 'H }      -- empty string
3698     ::= { jmGeneralEntry 7 }
3699
3700
3701
3702
3703
```

```

3704 -- The Job ID Group (MANDATORY)
3705
3706 -- The jmJobIDGroup consists entirely of the jmJobIDTable.
3707
3708 jmJobID OBJECT IDENTIFIER ::= { jobmonMIBObjects 2 }
3709
3710 jmJobIDTable OBJECT-TYPE
3711     SYNTAX      SEQUENCE OF JmJobIDEntry
3712     MAX-ACCESS  not-accessible
3713     STATUS      current
3714     DESCRIPTION
3715         "The jmJobIDTable provides a correspondence map (1) between the
3716         job submission ID that a client uses to refer to a job and (2)
3717         the jmGeneralJobSetIndex and jmJobIndex that the Job Monitoring
3718         MIB agent assigned to the job and that are used to access the
3719         job in all of the other tables in the MIB.  If a monitoring
3720         application already knows the jmGeneralJobSetIndex and the
3721         jmJobIndex of the job it is querying, that application NEED NOT
3722         use the jmJobIDTable.
3723
3724         The MANDATORY-GROUP macro specifies that this group is
3725         MANDATORY."
3726     ::= { jmJobID 1 }
3727
3728
3729
3730 jmJobIDEntry OBJECT-TYPE
3731     SYNTAX      JmJobIDEntry
3732     MAX-ACCESS  not-accessible
3733     STATUS      current
3734     DESCRIPTION
3735         "The map from (1) the jmJobSubmissionID to (2) the
3736         jmGeneralJobSetIndex and jmJobIndex.
3737
3738         An entry SHALL exist in this table for each job currently known
3739         to the agent for all job sets and job states.  There MAY be
3740         more than one jmJobIDEntry that maps to a single job.  This
3741         many to one mapping can occur when more than one network entity
3742         along the job submission path supplies a job submission ID.
3743         See Section 3.5.  However, each job SHALL appear once and in
3744         one and only one job set."
3745     INDEX { jmJobSubmissionID }
3746     ::= { jmJobIDTable 1 }
3747
3748 JmJobIDEntry ::= SEQUENCE {
3749     jmJobSubmissionID          OCTET STRING(SIZE(48)),
3750     jmJobIDJobSetIndex        Integer32 (0..32767),
3751     jmJobIDJobIndex           Integer32 (0..2147483647)
3752 }
3753

```

3754 jmJobSubmissionID OBJECT-TYPE
3755 SYNTAX OCTET STRING(SIZE(48))
3756 MAX-ACCESS not-accessible
3757 STATUS current
3758 DESCRIPTION
3759 "A quasi-unique 48-octet fixed-length string ID which
3760 identifies the job within a particular client-server
3761 environment. There are multiple formats for the
3762 jmJobSubmissionID. Each format SHALL be uniquely identified.
3763 See the JmJobSubmissionIDTypeTC textual convention. Each
3764 format SHALL be registered using the procedures of a type 2
3765 enum. See section 3.7.3 entitled: 'PWG Registration of Job
3766 Submission Id Formats'.
3767
3768 If the requester (client or server) does not supply a job
3769 submission ID in the job submission protocol, then the
3770 recipient (server or device) SHALL assign a job submission ID
3771 using any of the standard formats that have been reserved for
3772 agents and adding the final 8 octets to distinguish the ID from
3773 others submitted from the same requester.
3774
3775 The monitoring application, whether in the client or running
3776 separately, MAY use the job submission ID to help identify
3777 which jmJobIndex was assigned by the agent, i.e., in which row
3778 the job information is in the other tables.
3779
3780 NOTE - fixed-length is used so that a management application
3781 can use a shortened GetNext varbind (in SNMPv1 and SNMPv2) in
3782 order to get the next submission ID, disregarding the remainder
3783 of the ID in order to access jobs independent of the trailing
3784 identifier part, e.g., to get all jobs submitted by a
3785 particular jmJobOwner or submitted from a particular MAC
3786 address.
3787
3788 See the JmJobSubmissionIDTypeTC textual convention.
3789 See APPENDIX B - Support of Job Submission Protocols."
3790 ::= { jmJobIDEntry 1 }
3791

```
3792 jmJobIDJobSetIndex OBJECT-TYPE
3793     SYNTAX      Integer32 (0..32767)
3794     MAX-ACCESS  read-only
3795     STATUS      current
3796     DESCRIPTION
3797         "This object contains the value of the jmGeneralJobSetIndex for
3798         the job with the jmJobSubmissionID value, i.e., the job set
3799         index of the job set in which the job was placed when that
3800         server or device accepted the job. This 16-bit value in
3801         combination with the jmJobIDJobIndex value permits the
3802         management application to access the other tables to obtain the
3803         job-specific objects for this job.
3804
3805         See jmGeneralJobSetIndex in the jmGeneralTable."
3806     DEFVAL      { 0 }      -- 0 indicates no job set index
3807     ::= { jmJobIDEntry 2 }
3808
3809
3810
3811 jmJobIDJobIndex OBJECT-TYPE
3812     SYNTAX      Integer32 (0..2147483647)
3813     MAX-ACCESS  read-only
3814     STATUS      current
3815     DESCRIPTION
3816         "This object contains the value of the jmJobIndex for the job
3817         with the jmJobSubmissionID value, i.e., the job index for the
3818         job when the server or device accepted the job. This value, in
3819         combination with the jmJobIDJobSetIndex value, permits the
3820         management application to access the other tables to obtain the
3821         job-specific objects for this job.
3822
3823         See jmJobIndex in the jmJobTable."
3824     DEFVAL      { 0 }      -- 0 indicates no jmJobIndex value.
3825     ::= { jmJobIDEntry 3 }
3826
3827
3828
3829
```

```

3830 -- The Job Group (MANDATORY)
3831
3832 -- The jmJobGroup consists entirely of the jmJobTable.
3833
3834 jmJob OBJECT IDENTIFIER ::= { jobmonMIBObjects 3 }
3835
3836 jmJobTable OBJECT-TYPE
3837     SYNTAX      SEQUENCE OF JmJobEntry
3838     MAX-ACCESS  not-accessible
3839     STATUS      current
3840     DESCRIPTION
3841         "The jmJobTable consists of basic job state and status
3842         information for each job in a job set that (1) monitoring
3843         applications need to be able to access in a single SNMP Get
3844         operation, (2) that have a single value per job, and (3) that
3845         SHALL always be implemented.
3846
3847         The MANDATORY-GROUP macro specifies that this group is
3848         MANDATORY."
3849     ::= { jmJob 1 }
3850
3851
3852
3853 jmJobEntry OBJECT-TYPE
3854     SYNTAX      JmJobEntry
3855     MAX-ACCESS  not-accessible
3856     STATUS      current
3857     DESCRIPTION
3858         "Basic per-job state and status information.
3859
3860         An entry SHALL exist in this table for each job, no matter what
3861         the state of the job is. Each job SHALL appear in one and only
3862         one job set.
3863
3864         See Section 3.2 entitled 'The Job Tables'."
3865     INDEX { jmGeneralJobSetIndex, jmJobIndex }
3866     ::= { jmJobTable 1 }
3867
3868 JmJobEntry ::= SEQUENCE {
3869     jmJobIndex          Integer32 (1..2147483647),
3870     jmJobState          JmJobStateTC,
3871     jmJobStateReasons1 JmJobStateReasons1TC,
3872     jmNumberOfInterveningJobs Integer32 (-2..2147483647),
3873     jmJobKOctetsPerCopyRequested Integer32 (-2..2147483647),
3874     jmJobKOctetsProcessed Integer32 (-2..2147483647),
3875     jmJobImpressionsPerCopyRequested Integer32 (-2..2147483647),
3876     jmJobImpressionsCompleted Integer32 (-2..2147483647),
3877     jmJobOwner          JmJobStringTC (SIZE(0..63))
3878 }
3879

```



```
3880 jmJobIndex OBJECT-TYPE
3881     SYNTAX      Integer32 (1..2147483647)
3882     MAX-ACCESS  not-accessible
3883     STATUS      current
3884     DESCRIPTION
3885         "The sequential, monotonically increasing identifier index for
3886         the job generated by the server or device when that server or
3887         device accepted the job. This index value permits the
3888         management application to access the other tables to obtain the
3889         job-specific row entries.
3890
3891         See Section 3.2 entitled 'The Job Tables and the Oldest Active
3892         and Newest Active Indexes'.
3893         See Section 3.5 entitled 'Job Identification'.
3894         See also
3895
3896         jmGeneralNewestActiveJobIndex for the largest value of
3897         jmJobIndex.
3898         See JmJobSubmissionIDTypeTC for a limit on the size of this
3899         index if the agent represents it as an 8-digit decimal number."
3900     ::= { jmJobEntry 1 }
3901
3902
3903
3904 jmJobState OBJECT-TYPE
3905     SYNTAX      JmJobStateTC
3906     MAX-ACCESS  read-only
3907     STATUS      current
3908     DESCRIPTION
3909         "The current state of the job (pending, processing, completed,
3910         etc.). Agents SHALL implement only those states which are
3911         appropriate for the particular implementation. However,
3912         management applications SHALL be prepared to receive all the
3913         standard job states.
3914
3915         The final value for this object SHALL be one of: completed,
3916         canceled, or aborted. The minimum length of time that the
3917         agent SHALL maintain MIB data for a job in the completed,
3918         canceled, or aborted state before removing the job data from
3919         the jmJobIDTable and jmJobTable is specified by the value of
3920         the jmGeneralJobPersistence object."
3921     DEFVAL      { unknown }      -- default is unknown
3922     ::= { jmJobEntry 2 }
3923
```

```
3924 jmJobStateReasons1 OBJECT-TYPE
3925     SYNTAX      JmJobStateReasons1TC
3926     MAX-ACCESS  read-only
3927     STATUS      current
3928     DESCRIPTION
3929         "Additional information about the job's current state, i.e.,
3930         information that augments the value of the job's jmJobState
3931         object.
3932
3933         Implementation of any reason values is OPTIONAL, but an agent
3934         SHOULD return any reason information available. These values
3935         MAY be used with any job state or states for which the reason
3936         makes sense. Since the Job State Reasons will be more dynamic
3937         than the Job State, it is recommended that a job monitoring
3938         application read this object every time jmJobState is read.
3939         When the agent cannot provide a reason for the current state of
3940         the job, the value of the jmJobStateReasons1 object and
3941         jobStateReasonsN attributes SHALL be 0.
3942
3943         The jobStateReasonsN (N=2..4) attributes provide further
3944         additional information about the job's current state."
3945     DEFVAL      { 0 }          -- no reasons
3946     ::= { jmJobEntry 3 }
3947
3948
3949
3950 jmNumberOfInterveningJobs OBJECT-TYPE
3951     SYNTAX      Integer32 (-2..2147483647)
3952     MAX-ACCESS  read-only
3953     STATUS      current
3954     DESCRIPTION
3955         "The number of jobs that are expected to complete processing
3956         before this job has completed processing according to the
3957         implementation's queuing algorithm, if no other jobs were to be
3958         submitted. In other words, this value is the job's queue
3959         position. The agent SHALL return a value of 0 for this
3960         attribute when the job is the next job to complete processing
3961         (or has completed processing)."
3962     DEFVAL      { 0 }          -- default is no intervening jobs.
3963     ::= { jmJobEntry 4 }
3964
```

```
3965 jmJobKOctetsPerCopyRequested OBJECT-TYPE
3966     SYNTAX      Integer32 (-2..2147483647)
3967     MAX-ACCESS  read-only
3968     STATUS      current
3969     DESCRIPTION
3970         "The total size in K (1024) octets of the document(s) being
3971         requested to be processed in the job.  The agent SHALL round
3972         the actual number of octets up to the next highest K.  Thus 0
3973         octets is represented as '0', 1-1024 octets is represented as
3974         '1', 1025-2048 is represented as '2', etc.
3975
3976         In computing this value, the server/device SHALL NOT include
3977         the multiplicative factors contributed by (1) the number of
3978         document copies, and (2) the number of job copies, independent
3979         of whether the device can process multiple copies of the job or
3980         document without making multiple passes over the job or
3981         document data and independent of whether the output is collated
3982         or not.  Thus the server/device computation is independent of
3983         the implementation and indicates the size of the document(s)
3984         measured in K octets independent of the number of copies."
3985     DEFVAL      { -2 }      -- the default is unknown(-2)
3986     ::= { jmJobEntry 5 }
```

```
3987
3988
3989
3990 jmJobKOctetsProcessed OBJECT-TYPE
3991     SYNTAX      Integer32 (-2..2147483647)
3992     MAX-ACCESS  read-only
3993     STATUS      current
3994     DESCRIPTION
3995         "The total number of octets processed by the server or device
3996         measured in units of K (1024) octets so far.  The agent SHALL
3997         round the actual number of octets processed up to the next
3998         higher K.  Thus 0 octets is represented as '0', 1-1024 octets
3999         is represented as '1', 1025-2048 octets is '2', etc.  For
4000         printing devices, this value is the number interpreted by the
4001         page description language interpreter rather than what has been
4002         marked on media.
4003
4004         For implementations where multiple copies are produced by the
4005         interpreter with only a single pass over the data, the final
4006         value SHALL be equal to the value of the
4007         jmJobKOctetsPerCopyRequested object.  For implementations where
4008         multiple copies are produced by the interpreter by processing
4009         the data for each copy, the final value SHALL be a multiple of
4010         the value of the jmJobKOctetsPerCopyRequested object.
4011
4012         NOTE - See the impressionsCompletedCurrentCopy and
4013         pagesCompletedCurrentCopy attributes for attributes that are
4014         reset on each document copy.
4015
```

4016 NOTE - The jmJobKOctetsProcessed object can be used with the
4017 jmJobKOctetsPerCopyRequested object to provide an indication of
4018 the relative progress of the job, provided that the
4019 multiplicative factor is taken into account for some
4020 implementations of multiple copies."
4021 DEFVAL { 0 } -- default is no octets processed.
4022 ::= { jmJobEntry 6 }
4023
4024
4025 jmJobImpressionsPerCopyRequested OBJECT-TYPE
4026 SYNTAX Integer32 (-2..2147483647)
4027 MAX-ACCESS read-only
4028 STATUS current
4029 DESCRIPTION
4030 "The total size in number of impressions of the document(s)
4031 submitted.
4032
4033 In computing this value, the server/device SHALL NOT include
4034 the multiplicative factors contributed by (1) the number of
4035 document copies, and (2) the number of job copies, independent
4036 of whether the device can process multiple copies of the job or
4037 document without making multiple passes over the job or
4038 document data and independent of whether the output is collated
4039 or not. Thus the server/device computation is independent of
4040 the implementation and reflects the size of the document(s)
4041 measured in impressions independent of the number of copies.
4042
4043 See the definition of the term 'impression' in Section 2."
4044 DEFVAL { -2 } -- default is unknown(-2)
4045 ::= { jmJobEntry 7 }
4046
4047
4048 jmJobImpressionsCompleted OBJECT-TYPE
4049 SYNTAX Integer32 (-2..2147483647)
4050 MAX-ACCESS read-only
4051 STATUS current
4052 DESCRIPTION
4053 "The total number of impressions completed for this job so far.
4054 For printing devices, the impressions completed includes
4055 interpreting, marking, and stacking the output. For other
4056 types of job services, the number of impressions completed
4057 includes the number of impressions processed.
4058
4059 NOTE - See the impressionsCompletedCurrentCopy and
4060 pagesCompletedCurrentCopy attributes for attributes that are
4061 reset on each document copy.
4062
4063 NOTE - The jmJobImpressionsCompleted object can be used with
4064 the jmJobImpressionsPerCopyRequested object to provide an
4065 indication of the relative progress of the job, provided that
4066 the multiplicative factor is taken into account for some
4067 implementations of multiple copies.

```
4068
4069     See the definition of the term 'impression' in Section 2 and
4070     the counting example in Section 3.4 entitled 'Monitoring Job
4071     Progress'."
4072     DEFVAL      { 0 }          -- default is no octets
4073     ::= { jmJobEntry 8 }
4074
4075
4076
4077 jmJobOwner OBJECT-TYPE
4078     SYNTAX      JmJobStringTC (SIZE(0..63))
4079     MAX-ACCESS  read-only
4080     STATUS      current
4081     DESCRIPTION
4082         "The coded character set name of the user that submitted the
4083         job.  The method of assigning this user name will be system
4084         and/or site specific but the method MUST ensure that the name
4085         is unique to the network that is visible to the client and
4086         target device.
4087
4088         This value SHOULD be the most authenticated name of the user
4089         submitting the job.
4090
4091         See the OBJECT compliance macro for the minimum maximum length
4092         required for conformance."
4093     DEFVAL      { ''H }          -- default is empty string
4094     ::= { jmJobEntry 9 }
4095
4096
4097
4098
```

```
4099 -- The Attribute Group (MANDATORY)
4100
4101 -- The jmAttributeGroup consists entirely of the jmAttributeTable.
4102 --
4103 -- Implementation of the objects in this group is MANDATORY.
4104 -- See Section 3.1 entitled 'Conformance Considerations'.
4105 -- An agent SHALL implement any attribute if (1) the server or device
4106 -- supports the functionality represented by the attribute and (2) the
4107 -- information is available to the agent.
4108
4109 jmAttribute OBJECT IDENTIFIER ::= { jobmonMIBObjects 4 }
4110
4111
4112
4113 jmAttributeTable OBJECT-TYPE
4114     SYNTAX          SEQUENCE OF JmAttributeEntry
4115     MAX-ACCESS     not-accessible
4116     STATUS         current
4117     DESCRIPTION
4118         "The jmAttributeTable SHALL contain attributes of the job and
4119         document(s) for each job in a job set.  Instead of allocating
4120         distinct objects for each attribute, each attribute is
4121         represented as a separate row in the jmAttributeTable.
4122
4123         The MANDATORY-GROUP macro specifies that this group is
4124         MANDATORY.  An agent SHALL implement any attribute if (1) the
4125         server or device supports the functionality represented by the
4126         attribute and (2) the information is available to the agent. "
4127     ::= { jmAttribute 1 }
4128
4129
4130
4131 jmAttributeEntry OBJECT-TYPE
4132     SYNTAX          JmAttributeEntry
4133     MAX-ACCESS     not-accessible
4134     STATUS         current
4135     DESCRIPTION
4136         "Attributes representing information about the job and
4137         document(s) or resources required and/or consumed.
4138
4139         Each entry in the jmAttributeTable is a per-job entry with an
4140         extra index for each type of attribute (jmAttributeTypeIndex)
4141         that a job can have and an additional index
4142         (jmAttributeInstanceIndex) for those attributes that can have
4143         multiple instances per job.  The jmAttributeTypeIndex object
4144         SHALL contain an enum type that indicates the type of attribute
4145         (see the JmAttributeTypeTC textual-convention).  The value of
4146         the attribute SHALL be represented in either the
4147         jmAttributeValueAsInteger or jmAttributeValueAsOctets objects,
4148         and/or both, as specified in the JmAttributeTypeTC textual-
4149         convention.
4150
```

4151 The agent SHALL create rows in the jmAttributeTable as the
 4152 server or device is able to discover the attributes either from
 4153 the job submission protocol itself or from the document PDL.
 4154 As the documents are interpreted, the interpreter MAY discover
 4155 additional attributes and so the agent adds additional rows to
 4156 this table. As the attributes that represent resources are
 4157 actually consumed, the usage counter contained in the
 4158 jmAttributeValueAsInteger object is incremented according to
 4159 the units indicated in the description of the JmAttributeTypeTC
 4160 enum.

4161
 4162 The agent SHALL maintain each row in the jmAttributeTable for
 4163 at least the minimum time after a job completes as specified by
 4164 the jmGeneralAttributePersistence object.

4165
 4166 Zero or more entries SHALL exist in this table for each job in
 4167 a job set.

4168
 4169 See Section 3.3 entitled 'The Attribute Mechanism' for a
 4170 description of the jmAttributeTable."

4171 INDEX { jmGeneralJobSetIndex, jmJobIndex, jmAttributeTypeIndex,
 4172 jmAttributeInstanceIndex }
 4173 ::= { jmAttributeTable 1 }

4174
 4175 JmAttributeEntry ::= SEQUENCE {
 4176 jmAttributeTypeIndex JmAttributeTypeTC,
 4177 jmAttributeInstanceIndex Integer32 (1..32767),
 4178 jmAttributeValueAsInteger Integer32 (-2..2147483647),
 4179 jmAttributeValueAsOctets OCTET STRING(SIZE(0..63))
 4180 }
 4181

```
4182 jmAttributeTypeIndex OBJECT-TYPE
4183     SYNTAX      JmAttributeTypeTC
4184     MAX-ACCESS  not-accessible
4185     STATUS      current
4186     DESCRIPTION
4187         "The type of attribute that this row entry represents.
4188
4189         The type MAY identify information about the job or document(s)
4190         or MAY identify a resource required to process the job before
4191         the job start processing and/or consumed by the job as the job
4192         is processed.
4193
4194         Examples of job attributes (i.e., apply to the job as a whole)
4195         that have only one instance per job include:
4196         jobCopiesRequested(90), documentCopiesRequested(92),
4197         jobCopiesCompleted(91), documentCopiesCompleted(93), while
4198         examples of job attributes that may have more than one instance
4199         per job include: documentFormatIndex(37), and
4200         documentFormat(38).
4201
4202         Examples of document attributes (one instance per document)
4203         include: fileName(34), and documentName(35).
4204
4205         Examples of required and consumed resource attributes include:
4206         pagesRequested(130), mediumRequested(170), pagesCompleted(131),
4207         and mediumConsumed(171), respectively."
4208 ::= { jmAttributeEntry 1 }
4209
4210
4211
4212 jmAttributeInstanceIndex OBJECT-TYPE
4213     SYNTAX      Integer32 (1..32767)
4214     MAX-ACCESS  not-accessible
4215     STATUS      current
4216     DESCRIPTION
4217         "A running 16-bit index of the attributes of the same type for
4218         each job.  For those attributes with only a single instance per
4219         job, this index value SHALL be 1.  For those attributes that
4220         are a single value per document, the index value SHALL be the
4221         document number, starting with 1 for the first document in the
4222         job.  Jobs with only a single document SHALL use the index
4223         value of 1.  For those attributes that can have multiple values
4224         per job or per document, such as documentFormatIndex(37) or
4225         documentFormat(38), the index SHALL be a running index for the
4226         job as a whole, starting at 1."
4227 ::= { jmAttributeEntry 2 }
4228
```



```
4229 jmAttributeValueAsInteger OBJECT-TYPE
4230     SYNTAX      Integer32 (-2..2147483647)
4231     MAX-ACCESS  read-only
4232     STATUS      current
4233     DESCRIPTION
4234         "The integer value of the attribute.  The value of the
4235         attribute SHALL be represented as an integer if the enum
4236         description in the JmAttributeTypeTC textual-convention
4237         definition has the tag: 'INTEGER:'.
```

4238

4239 Depending on the enum definition, this object value MAY be an
4240 integer, a counter, an index, or an enum, depending on the
4241 jmAttributeTypeIndex value. The units of this value are
4242 specified in the enum description.

4243

4244 For those attributes that are accumulating job consumption as
4245 the job is processed as specified in the JmAttributeTypeTC
4246 textual-convention, SHALL contain the final value after the job
4247 completes processing, i.e., this value SHALL indicate the total
4248 usage of this resource made by the job.

4249

4250 A monitoring application is able to copy this value to a
4251 suitable longer term storage for later processing as part of an
4252 accounting system.

4253

4254 Since the agent MAY add attributes representing resources to
4255 this table while the job is waiting to be processed or being
4256 processed, which can be a long time before any of the resources
4257 are actually used, the agent SHALL set the value of the
4258 jmAttributeValueAsInteger object to 0 for resources that the
4259 job has not yet consumed.

4260

4261 Attributes for which the concept of an integer value is
4262 meaningless, such as fileName(34), jobName, and
4263 processingMessage, do not have the 'INTEGER:' tag in the
4264 JmAttributeTypeTC definition and so an agent SHALL always
4265 return a value of '-1' to indicate 'other' for the value of the
4266 jmAttributeValueAsInteger object for these attributes.

4267

4268 For attributes which do have the 'INTEGER:' tag in the
4269 JmAttributeTypeTC definition, if the integer value is not (yet)
4270 known, the agent either (1) SHALL not materialize the row in
4271 the jmAttributeTable until the value is known or (2) SHALL
4272 return a '-2' to represent an 'unknown' counting integer value,
4273 a '0' to represent an 'unknown' index value, and a '2' to
4274 represent an 'unknown(2)' enum value."

```
4275     DEFVAL      { -2 }      -- default value is unknown(-2)
4276     ::= { jmAttributeEntry 3 }
```

4277

```
4278 jmAttributeValueAsOctets OBJECT-TYPE
4279     SYNTAX      OCTET STRING(SIZE(0..63))
4280     MAX-ACCESS  read-only
4281     STATUS      current
4282     DESCRIPTION
4283         "The octet string value of the attribute.  The value of the
4284         attribute SHALL be represented as an OCTET STRING if the enum
4285         description in the JmAttributeTypeTC textual-convention
4286         definition has the tag: 'OCTETS:'."
4287
4288         Depending on the enum definition, this object value MAY be a
4289         coded character set string (text), such as 'JmUTF8StringTC', or
4290         a binary octet string, such as 'DateAndTime'.
4291
4292         Attributes for which the concept of an octet string value is
4293         meaningless, such as pagesCompleted, do not have the tag
4294         'OCTETS:' in the JmAttributeTypeTC definition and so the agent
4295         SHALL always return a zero length string for the value of the
4296         jmAttributeValueAsOctets object.
4297
4298         For attributes which do have the 'OCTETS:' tag in the
4299         JmAttributeTypeTC definition, if the OCTET STRING value is not
4300         (yet) known, the agent either SHALL NOT materialize the row in
4301         the jmAttributeTable until the value is known or SHALL return a
4302         zero-length string."
4303     DEFVAL      { ''H } -- empty string
4304     ::= { jmAttributeEntry 4 }
4305
```

4306
4307 -- The Mirror Attribute Group (OPTIONAL)
4308
4309 -- The jmMirrorAttrGroup consists entirely of the jmMirrorAttrTable.
4310 --
4311 -- Implementation of the objects in this group is OPTIONAL.
4312 -- See Section 3.1 entitled 'Conformance Considerations'.
4313 -- The jmMirrorAttrTable complements the MANDATORY jmAttributeTable.
4314 --
4315 -- The jmMirrorAttrTable provides access to all of the attributes that
4316 -- an implementation supports, sorted by attribute type (traditional
4317 -- SNMP MIB access), rather than being sorted by job set and job index
4318 -- (modern object-oriented access) as in the analogous
4319 -- jmAttributeTable.
4320
4321 jmMirrorAttr OBJECT IDENTIFIER ::= { jobmonMIBObjects 5 }
4322
4323 jmMirrorAttrTable OBJECT-TYPE
4324 SYNTAX SEQUENCE OF JmAttributeEntry
4325 MAX-ACCESS not-accessible
4326 STATUS current
4327 DESCRIPTION
4328 "The jmMirrorAttrTable is an OPTIONAL table which provides
4329 identical attributes to the jmAttributeTable but with a
4330 different index structure. See jmAttributeTable for further
4331 details.
4332
4333 See Section 3.3 entitled 'The Attribute Mechanism' for a
4334 description of the jmMirrorAttrTable."
4335 ::= { jmMirror 1 }
4336
4337
4338
4339 jmMirrorAttrEntry OBJECT-TYPE
4340 SYNTAX JmMirrorAttrEntry
4341 MAX-ACCESS not-accessible
4342 STATUS current
4343 DESCRIPTION
4344 "The attributes that represent information about each job and
4345 documents or resources required and/or consumed.
4346
4347 Each entry in jmMirrorAttrTable is a per-attribute entry with a
4348 primary index for each type of attribute (jmMirrorAttrTypeIndex)
4349 that a job can have and secondary indices which specify job set
4350 (jmJobSetIndex), job instance (jmJobIndex), and attribute
4351 instance (jmMirrorAttrInstanceIndex).
4352
4353 An agent which implements the jmMirrorAttrTable SHALL create
4354 and maintain a row in the jmMirrorAttrTable for each
4355 corresponding row in the jmAttributeTable."
4356 INDEX { jmMirrorAttrTypeIndex, jmGeneralJobSetIndex, jmJobIndex,
4357 jmMirrorAttrInstanceIndex }

```
4358     ::= { jmMirrorAttrTable 1 }
4359
4360 JmMirrorAttrEntry ::= SEQUENCE {
4361     jmMirrorAttrTypeIndex      JmAttributeTypeTC,
4362     jmMirrorAttrInstanceIndex  Integer32 (1..32767),
4363     jmMirrorAttrValueAsInteger Integer32 (-2..2147483647),
4364     jmMirrorAttrValueAsOctets  OCTET STRING(SIZE(0..63))
4365 }
4366
4367 jmMirrorAttrTypeIndex OBJECT-TYPE
4368     SYNTAX      JmAttributeTypeTC
4369     MAX-ACCESS  not-accessible
4370     STATUS      current
4371     DESCRIPTION
4372         "The type of attribute that this row entry represents.
4373
4374         See jmAttributeTypeIndex in jmAttributeTable for complete
4375         description."
4376     ::= { jmMirrorAttrEntry 1 }
4377
4378 jmMirrorAttrInstanceIndex OBJECT-TYPE
4379     SYNTAX      Integer32 (1..32767)
4380     MAX-ACCESS  not-accessible
4381     STATUS      current
4382     DESCRIPTION
4383         "The instance of attribute that this row entry represents.
4384
4385         See jmAttributeInstanceIndex in jmAttributeTable for complete
4386         description."
4387     ::= { jmMirrorAttrEntry 2 }
4388
4389 jmMirrorAttrValueAsInteger OBJECT-TYPE
4390     SYNTAX      Integer32 (-2..2147483647)
4391     MAX-ACCESS  read-only
4392     STATUS      current
4393     DESCRIPTION
4394         "The integer value of the attribute.
4395
4396         See jmAttributeValueAsInteger in jmAttributeTable for complete
4397         description."
4398     DEFVAL     { -2 }          -- default value is unknown(-2)
4399     ::= { jmMirrorAttrEntry 3 }
4400
4401 jmMirrorAttrValueAsOctets OBJECT-TYPE
4402     SYNTAX      OCTET STRING(SIZE(0..63))
4403     MAX-ACCESS  read-only
4404     STATUS      current
4405     DESCRIPTION
4406         "The octet string value of the attribute.
4407
4408         See jmAttributeValueAsOctets in jmAttributeTable for complete
4409         description."
```

```
4410     DEFVAL      { ''H }      -- empty string
4411     ::= { jmMirrorAttrEntry 4 }
```

```
4412 -- Notifications and Trapping
4413 -- Reserved for the future
4414
4415 jobmonMIBNotifications OBJECT IDENTIFIER ::= { jobmonMIB 2 }
4416
4417
4418
4419 -- Conformance Information
4420
4421 jmMIBConformance OBJECT IDENTIFIER ::= { jobmonMIB 3 }
4422
4423
4424
4425 -- compliance statements
4426 jmMIBCompliance MODULE-COMPLIANCE
4427     STATUS current
4428     DESCRIPTION
4429         "The compliance statement for agents that implement the
4430         job monitoring MIB."
4431     MODULE -- this module
4432     MANDATORY-GROUPS {
4433         jmGeneralGroup, jmJobIDGroup, jmJobGroup, jmAttributeGroup }
4434
4435     GROUP jmMirrorAttrGroup
4436     DESCRIPTION
4437         "The mirror attribute group (sorted by attribute type).
4438         Implementation of this group is OPTIONAL.
4439
4440         An agent that implements the jmMirrorAttrTable SHALL create and
4441         maintain for the same time a row in the jmMirrorAttrTable for
4442         each corresponding row in the jmAttributeTable."
4443
4444     OBJECT jmGeneralJobSetName
4445     SYNTAX JmUTF8StringTC (SIZE(0..8))
4446     DESCRIPTION
4447         "Only 8 octets maximum string length NEED be supported by the
4448         agent."
4449
4450     OBJECT jmJobOwner
4451     SYNTAX JmJobStringTC (SIZE(0..16))
4452     DESCRIPTION
4453         "Only 16 octets maximum string length NEED be supported by the
4454         agent."
4455
4456 -- There are no CONDITIONALLY MANDATORY or OPTIONAL groups.
4457
4458 ::= { jmMIBConformance 1 }
4459
```

```
4460 jmMIBGroups      OBJECT IDENTIFIER ::= { jmMIBConformance 2 }
4461
4462 jmGeneralGroup OBJECT-GROUP
4463     OBJECTS {
4464         jmGeneralNumberOfActiveJobs,    jmGeneralOldestActiveJobIndex,
4465         jmGeneralNewestActiveJobIndex,  jmGeneralJobPersistence,
4466         jmGeneralAttributePersistence,  jmGeneralJobSetName}
4467     STATUS current
4468     DESCRIPTION
4469         "The general group."
4470     ::= { jmMIBGroups 1 }
4471
4472
4473
4474 jmJobIDGroup OBJECT-GROUP
4475     OBJECTS {
4476         jmJobIDJobSetIndex, jmJobIDJobIndex }
4477     STATUS current
4478     DESCRIPTION
4479         "The job ID group."
4480     ::= { jmMIBGroups 2 }
4481
4482
4483
4484 jmJobGroup OBJECT-GROUP
4485     OBJECTS {
4486         jmJobState, jmJobStateReasons1, jmNumberOfInterveningJobs,
4487         jmJobKOctetsPerCopyRequested, jmJobKOctetsProcessed,
4488         jmJobImpressionsPerCopyRequested, jmJobImpressionsCompleted,
4489         jmJobOwner }
4490     STATUS current
4491     DESCRIPTION
4492         "The job group."
4493     ::= { jmMIBGroups 3 }
4494
4495
4496
4497 jmAttributeGroup OBJECT-GROUP
4498     OBJECTS {
4499         jmAttributeValueAsInteger, jmAttributeValueAsOctets }
4500     STATUS current
4501     DESCRIPTION
4502         "The attribute group."
4503     ::= { jmMIBGroups 4 }
4504
4505
4506
4506 jmMirrorAttrGroup OBJECT-GROUP
4507     OBJECTS {
4508         jmMirrorAttrValueAsInteger, jmMirrorAttrValueAsOctets }
4509     STATUS current
4510     DESCRIPTION
```

```
4511         "The mirror attribute group (sorted by attribute type).
4512         Implementation of this group is OPTIONAL.
4513
4514         An agent which implements the jmMirrorAttrTable SHALL create
4515         and maintain for the same time a row in the jmMirrorAttrTable
4516         for each corresponding row in the jmAttributeTable."
4517 ::= { jmMIBGroups 5 }
4518
4519
4520 END
```


4521 5 Appendix A - Implementing the Job Life Cycle

4522 The job object has well-defined states and client operations that
4523 affect the transition between the job states. Internal server and
4524 device actions also affect the transitions of the job between the job
4525 states. These states and transitions are referred to as the job's *life*
4526 *cycle*.

4527 Not all implementations of job submission protocols have all of the
4528 states of the job model specified here. The job model specified here
4529 is intended to be a superset of most implementations. It is the
4530 purpose of the agent to map the particular implementation's job life
4531 cycle onto the one specified here. The agent MAY omit any states not
4532 implemented. Only the processing and completed states are required to
4533 be implemented by an agent. However, a conforming management
4534 application SHALL be prepared to accept any of the states in the job
4535 life cycle specified here, so that the management application can
4536 interoperate with any conforming agent.

4537 The job states are intended to be user visible. The agent SHALL make
4538 these states visible in the MIB, but only for the subset of job states
4539 that the implementation has. Some implementations MAY need to have
4540 sub-states of these user-visible states. The jmJobStateReasons1 object
4541 and the jobStateReasonsN ($N=2..4$) attributes can be used to represent
4542 the sub-states of the jobs.

4543 Job states are intended to last a user-visible length of time in most
4544 implementations. However, some jobs may pass through some states in
4545 zero time in some situations and/or in some implementations.

4546 The job model does not specify how accounting and auditing is
4547 implemented, except to assume that accounting and auditing logs are
4548 separate from the job life cycle and last longer than job entries in
4549 the MIB. Jobs in the completed, aborted, or canceled states are not
4550 logs, since jobs in these states are accessible via SNMP protocol
4551 operations and SHALL be removed from the Job Monitoring MIB tables
4552 after a site-settable or implementation-defined period of time. An
4553 accounting application MAY copy accounting information incrementally to
4554 an accounting log as a job processes, or MAY be copied while the job is
4555 in the canceled, aborted, or completed states, depending on
4556 implementation. The same is true for auditing logs.

4557 The jmJobState object specifies the standard job states. The normal
4558 job state transitions are shown in the state transition diagram
4559 presented in Table 1.

4560 6 APPENDIX B - Support of Job Submission Protocols

4561 A companion PWG document, entitled "Job Submission Protocol Mapping
4562 Recommendations for the Job Monitoring MIB" [protomap] contains the
4563 recommended usage of each of the objects and attributes in this MIB
4564 with a number of job submission protocols. In particular, which job
4565 submission ID format should be used is indicated for each job
4566 submission protocol.

4567 Some job submission protocols have support for the client to specify a
4568 job submission ID. A second approach is to enhance the document format
4569 to embed the job submission ID in the document data. This second
4570 approach is independent of the job submission protocol. This appendix
4571 lists some examples of these approaches.

4572 Some PJL implementations wrap a banner page as a PJL job around a job
4573 submitted by a client. If this results in multiple job submission IDs,
4574 the agent SHALL create multiple jmJobIDEntry rows in the jmJobIDTable
4575 that each point to the same job entry in the job tables. See the
4576 specification of the jmJobIDEntry.

4577 7 References

4578 [char-set-policy] Harald Avelstrand, "IETF Policy on Character Sets and
4579 Language", June 1997. Latest draft: <draft-avelstrand-charset-
4580 policy-00.txt>

4581 [GB2312] GB 2312-1980, "Chinese People's Republic of China (PRC) mixed
4582 one byte and two byte coded character set"

4583 [hr-mib] P. Grillo, S. Waldbusser, "Host Resources MIB", RFC 1514,
4584 September 1993

4585 [iana] J. Reynolds, and J. Postel, "Assigned Numbers", STD 2, RFC 1700,
4586 ISI, October 1994.

4587 [IANA-charsets] Coded Character Sets registered by IANA and assigned an
4588 enum value for use in the CodedCharSet textual convention imported from
4589 the Printer MIB. See ftp://ftp.isi.edu/in-
4590 notes/iana/assignments/character-sets

4591 [iana-media-types] IANA Registration of MIME media types (MIME content
4592 types/subtypes). See ftp://ftp.isi.edu/in-notes/iana/assignments/

4593 [ipp-model] Internet Printing Protocol/1.0: Model and Semantics, work
4594 in progress on the IETF standards track. See draft-ietf-ipp-model-
4595 09.txt. See also <http://www.pwg.org/ipp/index.html>

4596 [ISO-639] ISO 639:1988 (E/F) - Code for Representation of names of
4597 languages - The International Organization for Standardization, 1st
4598 edition, 1988.

4599 [ISO 646] ISO/IEC 646:1991, "Information technology -- ISO 7-bit coded
4600 character set for information interchange", JTC1/SC2.

4601 [ISO 8859] ISO/IEC 8859-1:1987, "Information technology -- 8-bit single
4602 byte coded graphic character sets - Part 1: Latin alphabet No. 1,
4603 JTC1/SC2."

4604 [ISO 2022] ISO/IEC 2022:1994 - "Information technology -- Character
4605 code structure and extension techniques", JTC1/SC2.

4606 [ISO-3166] ISO 3166:1988 (E/F) - Codes for representation of names of
4607 countries - The International Organization for Standardization, 3rd
4608 edition, 1988-08-15."

4609 [ISO-10646] ISO/IEC 10646-1:1993, "Information technology -- Universal
4610 Multiple-Octet Coded Character Set (UCS) - Part 1: Architecture and
4611 Basic Multilingual Plane, JTC1/SC2.

4612 [iso-dpa] ISO/IEC 10175 Document Printing Application (DPA). See
4613 <ftp://ftp.pwg.org/pub/pwg/dpa/>

4614 [JIS X0208] JIS X0208-1990, "Japanese two byte coded character set."

4615 [mib-II] MIB-II, RFC 1213.

4616 [print-mib] Smith, R., Wright, F., Hastings, T., Zilles, S. and
4617 Gyllenskog, J., "Printer MIB", RFC 1759, proposed IETF standard, March
4618 1995. See also [print-mib-draft].

4619 [print-mib-draft] Turner, R., "Printer MIB", work in progress, on the
4620 standards track as a draft standard: <draft-ietf-printmib-mib-info-
4621 02.txt>, October 15, 1997.

4622 [protomap] Bergman, R., "Job Submission Protocol Mapping
4623 Recommendations for the Job Monitoring MIB," work in progress as an
4624 informational RFC. See <draft-bergman-printmib-job-protomap-01.txt>,
4625 January 12, 1998.

4626 [pwg] The Printer Working Group is a printer industry consortium open
4627 to any individuals. For more information, access the PWG web page:
4628 <http://www.pwg.org>

4629 [req-words] S. Bradner, "Keywords for use in RFCs to Indicate
4630 Requirement Levels", RFC 2119, March 1997.

4631 [rfc 1738] Berners-Lee, T., Masinter, L., McCahill, M., "Uniform
4632 Resource Locators (URL)", RFC 1738, December 1994.

4633 [RFC-1766] Avelstrand, H., "Tags for the Identification of Languages",
4634 RFC 1766, March 1995.

4635 [rfc 2130] C. Weider, C. Preston, K. Simonsen, H. Alvestrand, R.
4636 Atkinson, M. Crispin, and P. Svanberg, "The Report of the IAB Character
4637 Set Workshop held 29 Feb-1 March, 1997", April 1997, RFC 2130.

4638 [SMIv2-SMI] J. Case, et al. "Structure of Management Information for
4639 Version 2 of the Simple Network Management Protocol (SNMPv2)", RFC
4640 1902, January 1996.

4641 [SMIv2-TC] J. Case, et al. "Textual Conventions for Version 2 of the
4642 Simple Network Management Protocol (SNMPv2)", RFC 1903, January 1996.

4643 [tipsi] IEEE 1284.1, Transport-independent Printer System Interface
4644 (TIPSI).

4645 [URI-spec] Berners-Lee, T., Masinter, L., McCahill, M. , "Uniform
4646 Resource Locators (URL)", RFC 1738, December, 1994.

4647 [US-ASCII] Coded Character Set - 7-bit American Standard Code for
4648 Information Interchange, ANSI X3.4-1986.

4649 [UTF-8] F. Yergeau, "UTF-8, a transformation format of Unicode and ISO
4650 10646", RFC 2044, October 1996.

4651 8 Author's Addresses

4652 Ron Bergman
4653 Dataproducts Corp.
4654 1757 Tapo Canyon Road
4655 Simi Valley, CA 93063-3394

4656
4657 Phone: 805-578-4421
4658 Fax: 805-578-4001
4659 Email: rbergman@dpc.com

4660
4661
4662 Tom Hastings
4663 Xerox Corporation, ESAE-231
4664 701 S. Aviation Blvd.
4665 El Segundo, CA 90245
4666
4667 Phone: 310-333-6413
4668 Fax: 310-333-5514
4669 EMail: hastings@cp10.es.xerox.com

4670
4671
4672 Scott A. Isaacson
4673 Novell, Inc.
4674 122 E 1700 S
4675 Provo, UT 84606
4676
4677 Phone: 801-861-7366
4678 Fax: 801-861-4025

4679 EEmail: scott_isaacson@novell.com

4680

4681

4682 Harry Lewis

4683 IBM Corporation

4684 6300 Diagonal Hwy

4685 Boulder, CO 80301

4686

4687 Phone: (303) 924-5337

4688 Fax:

4689 Email: harryl@us.ibm.com

4690

4691

4692 Send questions and comments to the Printer Working Group (PWG)
4693 using the Job Monitoring Project (JMP) Mailing List: jmp@pwg.org

4694

4695 To learn how to subscribe, send email to: jmp-request@pwg.org

4696

4697 Implementers of this specification are encouraged to join the jmp
4698 mailing list in order to participate in discussions on any
4699 clarifications needed and registration proposals for additional
4700 attributes and values being reviewed in order to achieve consensus.

4701

4702 For further information, access the PWG web page under "JMP":

4703

4704 <http://www.pwg.org/>

4705

4706 Other Participants:

4707 Chuck Adams - Tektronix

4708 Jeff Barnett - IBM

4709 Keith Carter, IBM Corporation

4710 Jeff Copeland - QMS

4711 Andy Davidson - Tektronix

4712 Roger deBry - IBM

4713 Mabry Dozier - QMS

4714 Lee Farrell - Canon

4715 Steve Gebert - IBM

4716 Robert Herriot - Sun Microsystems Inc.

4717 Shige Kanemitsu - Kyocera

4718 David Kellerman - Northlake Software

4719 Rick Landau - Digital

4720 Pete Loya - HP

4721 Ray Lutz - Cognisys

4722 Jay Martin - Underscore

4723 Mike MacKay, Novell, Inc.

4724 Stan McConnell - Xerox

4725 Carl-Uno Manros, Xerox, Corp.

4726 Pat Nogay - IBM

4727 Bob Pentecost - HP

4728 Rob Rhoads - Intel

4729 David Roach - Unisys
4730 Stuart Rowley - Kyocera
4731 Hiroyuki Sato - Canon
4732 Bob Setterbo - Adobe
4733 Gail Songer, EFI
4734 Mike Timperman - Lexmark
4735 Randy Turner - Sharp
4736 William Wagner - Digital Products
4737 Jim Walker - Dazel
4738 Chris Wellens - Interworking Labs
4739 Rob Whittle - Novell
4740 Don Wright - Lexmark
4741 Lloyd Young - Lexmark
4742 Atsushi Yuki - Kyocera
4743 Peter Zehler, Xerox, Corp.

4744 9 Change History

4745 This section summarizes the changes in each version after version 1.0
4746 in reverse chronological order.

4747 9.1 Changes to produce version 1.3, dated November 8, 1998

4748 The following changes were made to version 1.2, dated October 2, 1998
4749 to make version 1.3, dated November 8, 1998:

4750 1. Added the Mirror table.

4751 2. Moved the JmJobSubmissionIDTypeTC, JmJobStateReasons1TC,
4752 JmJobStateReasons2TC, JmJobStateReasons3TC, and JmJobStateReasons4TC
4753 assignments out of the MIB and into the Introduction.

4754

4755 9.2 Changes to produce version 1.2, dated October 2, 1998

4756 The following changes were made to version 1.1, dated October 1, 1998
4757 to make version 1.2, dated October 2, 1998:

4758 1. Removed all REFERENCE clauses since they referred to sections in the
4759 specification that were not in the MIB.

4760 2. Moved the definitions of the attributes from the TC to a new section
4761 3.3.8.

4762 3. Removed the attributes from the Table of Contents

4763 4. Added the data types as ASN.1 comments after each attribute enum.

4764 5. Changed a number of occurrences of "SHALL" to "is" when they were
4765 just definitions, rather than conformance requirements.

- 4766
- 4767 9.3 Changes to produce version 1.1, dated October 1, 1998
- 4768 The following changes were made to version 1.0, dated February 3, 1998
4769 to make version 1.1, dated October 1, 1998:
- 4770 1. Clarified sections 3.3.3 and 3.3.7 so that the DEFVAL of 0 for index
4771 attributes is different from the DEFVAL for
4772 jmAttributeValueAsInteger which is -2.
 - 4773 2. Clarified the relationships of the values of the
4774 JmJobCollationTypeTC with the IPP "multiple-document-handling"
4775 attribute.
 - 4776 3. Clarified that the values of the mediumRequested(170) and
4777 mediumConsumed(171) attributes may be any of the IPP 'media' values
4778 which are media names, media size names, and input tray names.
 - 4779 4. Added the two attributes approved by the PWG for registration in
4780 April 1998: mediumTypeConsumed(174) and mediumSizeConsumed(175).
 - 4781 5. Changed "insure" to "ensure".
 - 4782 6. Correct an incorrect reference in the jmAttributeEntry DESCRIPTION
4783 from jmJobTable to jmAttributeTable.

4784 10 INDEX

4785 This index includes the textual conventions, the objects, and the
4786 attributes. Textual conventions all start with the prefix: "JM" and
4787 end with the suffix: "TC". Objects all starts with the prefix: "jm"
4788 followed by the group name. Attributes are identified with enums, and
4789 so start with any lower case letter and have no special prefix.

4790
4791 colorantConsumed, 42
4792 colorantRequested, 41
4793 deviceNameRequested, 31
4794 documentCopiesCompleted, 36
4795 documentCopiesRequested, 36
4796 documentFormat, 33
4797 documentFormatIndex, 32
4798 documentName, 32
4799 fileName, 32
4800 finishing, 35
4801 fullColorImpressionsCompleted, 38
4802 highlightColorImpressionsCompleted, 39
4803 impressionsCompletedCurrentCopy, 38
4804 impressionsInterpreted, 38
4805 impressionsSentToDevice, 38
4806 impressionsSpooled, 38
4807 jmAttributeInstanceIndex, 104
4808 jmAttributeTypeIndex, 104
4809 JmAttributeTypeTC, 81
4810 jmAttributeValueAsInteger, 105
4811 jmAttributeValueAsOctets, 106
4812 JmBooleanTC, 75
4813 JmFinishingTC, 73
4814 jmGeneralAttributePersistence, 91
4815 jmGeneralJobPersistence, 91
4816 jmGeneralJobSetIndex, 89
4817 jmGeneralJobSetName, 92
4818 jmGeneralNewestActiveJobIndex, 90
4819 jmGeneralNumberOfActiveJobs, 89
4820 jmGeneralOldestActiveJobIndex, 90
4821 JmJobCollationTypeTC, 77
4822 jmJobIDJobIndex, 95
4823 jmJobIDJobSetIndex, 95
4824 jmJobImpressionsCompleted, 100
4825 jmJobImpressionsPerCopyRequested, 100
4826 jmJobIndex, 97
4827 jmJobKOctetsPerCopyRequested, 99
4828 jmJobKOctetsProcessed, 99
4829 jmJobOwner, 101
4830 JmJobServiceTypesTC, 85
4831 JmJobSourcePlatformTypeTC, 72
4832 jmJobState, 97
4833 jmJobStateReasons1, 98

4834 JmJobStateReasons1TC, 86
4835 JmJobStateReasons2TC, 86
4836 JmJobStateReasons3TC, 87
4837 JmJobStateReasons4TC, 87
4838 JmJobStateTC, 78
4839 JmJobStringTC, 71
4840 jmJobSubmissionID, 94
4841 JmJobSubmissionIDTypeTC, 77
4842 JmMediumTypeTC, 75
4843 jmMirrorAttrInstanceIndex, 108
4844 jmMirrorAttrTypeIndex, 108
4845 jmMirrorAttrValueAsInteger, 108
4846 jmMirrorAttrValueAsOctets, 108
4847 JmNaturalLanguageTagTC, 71
4848 jmNumberOfInterveningJobs, 98
4849 JmPrinterResolutionTC, 74
4850 JmPrintQualityTC, 74
4851 JmTimeStampTC, 71
4852 JmTonerEconomyTC, 75
4853 JmUTF8StringTC, 71
4854 jobAccountName, 28
4855 jobCodedCharSet, 27
4856 jobCollationType, 37
4857 jobComment, 32
4858 jobCompletionTime, 44
4859 jobCopiesCompleted, 36
4860 jobCopiesRequested, 36
4861 jobHold, 34
4862 jobHoldUntil, 34
4863 jobKOctetsTransferred, 37
4864 jobName, 29
4865 jobNaturalLanguageTag, 28
4866 jobOriginatingHost, 31
4867 jobPriority, 33
4868 jobProcessAfterDateAndTime, 34
4869 jobProcessingCPUtime, 44
4870 jobServiceTypes, 30
4871 jobSourceChannelIndex, 30
4872 jobSourcePlatformType, 30
4873 jobStartedBeingHeldTime, 43
4874 jobStartedProcessingTime, 44
4875 jobStateReasons2, 26
4876 jobStateReasons3, 26
4877 jobStateReasons4, 26
4878 jobSubmissionTime, 43
4879 jobSubmissionToServerTime, 43
4880 jobURI, 28
4881 mediumConsumed, 41
4882 mediumRequested, 41
4883 mediumSizeConsumed, 42
4884 mediumTypeConsumed, 42
4885 numberOfDocuments, 31

4886 other, 25
4887 outputBin, 34
4888 pagesCompleted, 39
4889 pagesCompletedCurrentCopy, 40
4890 pagesRequested, 39
4891 physicalDevice, 31
4892 printerResolutionRequested, 35
4893 printerResolutionUsed, 35
4894 printQualityRequested, 35
4895 printQualityUsed, 35
4896 processingMessage, 26
4897 processingMessageNaturalLangTag, 27
4898 queueNameRequested, 31
4899 serverAssignedJobName, 29
4900 sheetCompletedCopyNumber, 37
4901 sheetCompletedDocumentNumber, 37
4902 sheetsCompleted, 40
4903 sheetsCompletedCurrentCopy, 40
4904 sheetsRequested, 40
4905 sides, 35
4906 submittingApplicationName, 30
4907 submittingServerName, 30
4908 tonerDensityRequested, 35
4909 tonerDensityUsed, 36
4910 tonerEcomonyRequested, 35
4911 tonerEcomonyUsed, 35

4912