

# Job Monitoring MIB, V0.854

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

From: Tom Hastings

Date: 0807/0824/97

Version: 0.854

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

Status: ~~Eighth~~Seventh draft MIB that incorporates the~~corresponds to~~ resolutions of issues 110 to 120 from the 8/8 JMP meeting~~editorial comments on V0.83 and changes to keep in alignment with IPP (printer resolution syntax)~~. See the change history in the separate file: changes.doc .pdf.

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

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

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

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



25 INTERNET-DRAFT

26

27

28

29

30

31

32

33

34

Ron Bergman  
Dataproducts Corp.  
Tom Hastings  
Xerox Corporation  
Scott Isaacson  
Novell, Inc.  
Harry Lewis  
IBM Corp.  
August 8, July 1997

35

**Job Monitoring MIB - V0.854**

36

**<draft-ietf-printmib-job-monitor-054.txt>**

37

**Expires Feb 8, 1997**

38

**39 Status of this Memo**

40

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

41

42

43

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

44

45

46

47

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

48

49

50

51

**Abstract**

52

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

53

54

55

56

57

58

59

60

61

62

**TABLE OF CONTENTS**

63 **1. INTRODUCTION..... 9**

64 **1.1 Types of Information in the MIB .....9**

65 **1.2 Types of Job Monitoring Applications .....10**

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

67 **2.1 System Configurations for the Job Monitoring MIB .....14**

68 2.1.1 Configuration 1 - client-printer .....14

69 2.1.2 Configuration 2 - client-server-printer - agent in the server .....15

70 2.1.3 Configuration 3 - client-server-printer - client monitors printer agent and server .....16

71 **3. MANAGED OBJECT USAGE..... 18**

72 **3.1 Conformance Considerations.....18**

73 3.1.1 Conformance Terminology .....18

74 3.1.2 Agent Conformance Requirements .....18

75 3.1.2.1 MIB II System Group objects .....19

76 3.1.2.2 MIB II Interface Group objects .....19

77 3.1.2.3 Printer MIB objects .....19

78 3.1.3 Job Monitoring Application Conformance Requirements .....19

79 **3.2 The Job Tables and the Oldest Active and Newest Active Indexes .....20**

80 **3.3 The Attribute Mechanism.....21**

81 3.3.1 Conformance of Attribute Implementation.....22

82 3.3.2 Useful, 'Unknown', and 'Other' Values for Objects and Attributes .....23

83 3.3.3 Data Sub-types and Attribute Naming Conventions .....23

84 3.3.4 Single-Value (Row) Versus Multi-Value (MULTI-ROW) Attributes.....24

85 3.3.5 Requested Attributes.....24

86 3.3.6 Consumption Attributes.....25

87 3.3.7 Index Value Attributes .....25

88 **3.4 Job Identification .....25**

89 **3.5 Internationalization Considerations .....26**

90 3.5.1 'JmUTF8StringTC' for text generated by the server or device.....25

91 3.5.2 'JmJobStringTC' for text generated by the job submitter.....25

92 3.5.3 'DateAndTime' for representing the date and time .....25

93 **3.6 IANA Considerations.....28**

94 3.6.1 IANA Registration of enums .....28

95	3.6.1.1 Type 1 enumerations.....	28
96	3.6.1.2 Type 2 enumerations.....	28
97	3.6.1.3 Type 3 enumeration.....	29
98	3.6.2 IANA Registration of type 2 bit values.....	29
99	3.6.3 IANA Registration of Job Submission Id Formats.....	29
100	3.6.4 IANA Registration of MIME types/sub-types for document-formats.....	29
101	<b>3.7 Security Considerations.....</b>	<b>30</b>
102	3.7.1 Read-Write objects.....	30
103	3.7.2 Read-Only Objects In Other User's Jobs.....	30
104	<b>3.8 Values for Objects.....</b>	<b>23</b>
105	<b>3.9 Notification-s.....</b>	<b>30</b>
106	<b>4. MIB SPECIFICATION.....</b>	<b>30</b>
107	<b>Textual conventions for this MIB module.....</b>	<b>32</b>
108	<u>JmUTF8StringTC</u> .....	33
109	<u>JmJobStringTC</u> .....	33
110	JmTimeStampTC—simple time in seconds.....	33
111	JmJobSourcePlatformTypeTC—operating system platform definitions.....	33
112	JmFinishingTC—device finishing definitions.....	34
113	JmPrintQualityTC—print quality.....	35
114	JmPrinterResolutionTC—printer resolution.....	36
115	JmTonerEconomyTC—toner economy setting.....	36
116	JmBooleanTC—Boolean value.....	36
117	JmMediumTypeTC—medium type definitions.....	37
118	JmJobSubmissionIDTypeTC—job submission ID type definitions.....	38
119	JmJobStateTC—job state definitions.....	40
120	JmAttributeTypeTC—attribute type definitions.....	42
121	other (Int32(-2..) and/or Octets63).....	43
122	Job State attributes.....	43
123	jobStateReasons2 (JmJobStateReasons2TC).....	43
124	jobStateReasons3 (JmJobStateReasons3TC).....	43
125	jobStateReasons4 (JmJobStateReasons4TC).....	44
126	processingMessage (UTF8StringOetets63).....	44
127	<u>jobCodedCharSet (CodedCharSet)</u> .....	44
128	Job Identification attributes.....	44
129	jobAccountName (JobStringOetets63).....	44
130	serverAssignedJobName (JobStringOetets63).....	45
131	jobName (JobStringOetets63).....	45
132	jobServiceTypes (JmJobServiceTypesTC).....	45
133	jobSourceChannelIndex (Int32(0..)).....	46
134	jobSourcePlatformType (JmJobSourcePlatformTypeTC).....	46
135	submittingServerName (JobStringOetets63).....	46
136	submittingApplicationName (JobStringOetets63).....	46
137	jobOriginatingHost (JobStringOetets63).....	46

138	deviceNameRequested ( <u>JobStringOctets63</u> ).....	46
139	queueNameRequested ( <u>JobStringOctets63</u> ).....	46
140	physicalDevice (hrDeviceIndex and/or <u>UTF8StringOctets63</u> ).....	47
141	numberOfDocuments (Int32(-2..)).....	47
142	fileName ( <u>JobStringOctets63</u> ).....	47
143	documentName ( <u>JobStringOctets63</u> ).....	47
144	jobComment ( <u>JobStringOctets63</u> ).....	47
145	documentFormatIndex (Int32(0..)).....	47
146	documentFormat (PrtInterpreterLangFamilyTC and/or Octets63).....	48
147	Job Parameter attributes.....	48
148	jobPriority (Int32(1..100)).....	48
149	jobProcessAfterDateAndTime (DateAndTime).....	48
150	jobHold (JmBooleanTC).....	49
151	jobHoldUntil ( <u>JobStringOctets63</u> ).....	49
152	outputBin (Int32(0..) and/or <u>JobStringOctets63</u> ).....	49
153	sides (Int32(-2..2)).....	49
154	finishing (JmFinishingTC).....	49
155	Image Quality attributes (requested and used).....	49
156	printQualityRequested (JmPrintQualityTC).....	49
157	printQualityUsed (JmPrintQualityTC).....	50
158	printerResolutionRequested (JmPrinterResolutionTC).....	50
159	printerResolutionUsed (JmPrinterResolutionTC).....	50
160	tonerEcomonyRequested (JmTonerEconomyTC).....	50
161	tonerEcomonyUsed (JmTonerEconomyTC).....	50
162	tonerDensityRequested (Int32(-2..100)).....	50
163	tonerDensityUsed (Int32(-2..100)).....	50
164	Job Progress attributes (requested and consumed).....	50
165	jobCopiesRequested (Int32(-2..)).....	50
166	jobCopiesCompleted (Int32(-2..)).....	50
167	documentCopiesRequested (Int32(-2..)).....	50
168	documentCopiesCompleted (Int32(-2..)).....	51
169	jobKOctetsTransferred (Int32(-2..)).....	51
170	Impression attributes (requested and consumed).....	51
171	impressionsSpooled (Int32(-2..)).....	51
172	impressionsSentToDevice (Int32(-2..)).....	51
173	impressionsInterpreted (Int32(-2..)).....	52
174	impressionsCompletedCurrentCopy (Int32(-2..)).....	52
175	fullColorImpressionsCompleted (Int32(-2..)).....	52
176	highlightColorImpressionsCompleted (Int32(-2..)).....	52
177	Page attributes (requested and consumed).....	52
178	pagesRequested (Int32(-2..)).....	52
179	pagesCompleted (Int32(-2..)).....	52
180	pagesCompletedCurrentCopy (Int32(-2..)).....	53
181	Sheet attributes (requested and consumed).....	53
182	sheetsRequested (Int32(-2..)).....	53
183	sheetsCompleted (Int32(-2..)).....	53
184	sheetsCompletedCurrentCopy (Int32(-2..)).....	53
185	Resource attributes (requested and consumed).....	53
186	mediumRequested (JmMediumTypeTC and/or <u>JobStringOctets63</u> ).....	53

187	mediumConsumed ( <u>JobStringOetets63</u> ).....	54
188	colorantRequested (Int32(-2..) and/or <u>JobStringOetets63</u> ) .....	54
189	colorantConsumed (Int32(-2..) and/or <u>JobStringOetets63</u> ).....	54
190	Time attributes (set by server or device).....	54
191	jobSubmissionToServerTime (JmTimeStampTC and/or DateAndTime).....	55
192	jobSubmissionTime (JmTimeStampTC and/or DateAndTime).....	55
193	jobStartedBeingHeldTime (JmTimeStampTC and/or DateAndTime) .....	55
194	jobStartedProcessingTime (JmTimeStampTC and/or DateAndTime) .....	55
195	jobCompletedTime (JmTimeStampTC and/or DateAndTime) .....	55
196	jobProcessingCPUTime (Int32(-2..)).....	56
197	JmJobServiceTypesTC—bit encoded job service type definitions.....	57
198	JmJobStateReasons1TC—additional information about job states .....	59
199	JmJobStateReasons2TC—More additional information about job states .....	62
200	JmJobStateReasons3TC—More additional information about job states .....	65
201	JmJobStateReasons4TC—More additional information about job states .....	65
202	<b>The General Group (MANDATORY) .....</b>	<b>67</b>
203	jmGeneralJobSetIndex (Int32(1..32767)).....	67
204	jmGeneralNumberOfActiveJobs (Int32(0..)).....	68
205	jmGeneralOldestActiveJobIndex (Int32(0..)).....	68
206	jmGeneralNewestActiveJobIndex (Int32(0..)).....	68
207	jmGeneralJobPersistence (Int32(15..)).....	69
208	jmGeneralAttributePersistence (Int32(15..)).....	69
209	jmGeneralJobSetName ( <u>UTF8StringOetets63</u> ) .....	69
210	<b>The Job ID Group (MANDATORY).....</b>	<b>70</b>
211	jmJobSubmissionID (OCTET STRING(SIZE(48))).....	71
212	jmJobIDJobSetIndex (Int32(1..32767)).....	71
213	jmJobIDJobIndex (Int32(1..)).....	71
214	<b>The Job Group (MANDATORY).....</b>	<b>72</b>
215	jmJobIndex (Int32(1..)).....	73
216	jmJobState (JmJobStateTC).....	73
217	jmJobStateReasons1 (JmJobStateReasons1TC).....	73
218	jmNumberOfInterveningJobs (Int32(-2..)).....	74
219	jmJobKOctetsRequested (Int32(-2..)).....	74
220	jmJobKOctetsProcessed (Int32(-2..)).....	74
221	jmJobImpressionsRequested (Int32(-2..)).....	75
222	jmJobImpressionsCompleted (Int32(-2..)).....	75
223	jmJobOwner ( <u>JobStringOetets63</u> ).....	76
224	<b>The Attribute Group (MANDATORY) .....</b>	<b>76</b>
225	jmAttributeTypeIndex (JmAttributeTypeTC).....	77
226	jmAttributeInstanceIndex (Int32(1..32767)).....	78
227	jmAttributeValueAsInteger (Int32(-2..)).....	78
228	jmAttributeValueAsOetets (Oetets63) .....	79
229	<b>5. APPENDIX A - IMPLEMENTING THE JOB LIFE CYCLE.....</b>	<b>82</b>

230 **6. APPENDIX B - SUPPORT OF THE JOB SUBMISSION ID IN JOB**  
231 **SUBMISSION PROTOCOLS ..... 82**

232     **6.1 Hewlett-Packard's Printer Job Language (PJL) .....83**  
233     **6.2 ISO DPA.....83**

234 **7. REFERENCES..... 83**

235 **8. AUTHOR'S ADDRESSES..... 84**

236 **9. INDEX ..... 88**  
237



238

**Job Monitoring MIB**239 **1. Introduction**

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

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

261 **1.1 Types of Information in the MIB**

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

264 User:

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

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

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

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

272 Operator:  
273 Provide a presentation of the state of all the jobs in the print system.  
274 Provide the ability to identify the user that submitted the print job.  
275 Provide the ability to identify the resources required by each job.  
276 Provide the ability to define which physical printers are candidates for the print  
277 job.  
278 Provide some idea of how long each job will take. However, exact estimates of  
279 time to process a job is not being attempted. Instead, objects are included that  
280 allow the operator to be able to make gross estimates.

281 Capacity Planner:  
282 Provide the ability to determine printer utilization as a function of time.  
283 Provide the ability to determine how long jobs wait before starting to print.

284 Accountant:  
285 Provide information to allow the creation of a record of resources consumed and  
286 printer usage data for charging users or groups for resources consumed.  
287 Provide information to allow the prediction of consumable usage and resource  
288 need.

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

## 294 **1.2 Types of Job Monitoring Applications**

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

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

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

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

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

## 330 2. Terminology and Job Model

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

333 NOTE - Existing systems use conflicting terms, so these terms are drawn from the ISO  
334 10175 Document Printing Application (DPA) standard[iso-dpa]. For example,  
335 PostScript systems use the term *session* for what is called a *job* in this specification and  
336 the term *job* to mean what is called a *document* in this specification. PjL systems use  
337 the term *job* to mean what is called a *job* in this specification. PjL also supports  
338 multiple *documents* per job, but does not support specifying per-document attributes  
339 independently for each document.

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

342 Job Set: a group of jobs that are queued and scheduled together according to a specified  
343 scheduling algorithm for a specified device or set of devices. For implementations that  
344 embed the SNMP agent in the device, the MIB job set normally represents *all* the jobs  
345 known to the device, so that the implementation only implements a single job set. If the

346 SNMP agent is implemented in a server that controls one or more devices, each MIB job  
347 set represents a job queue for (1) a specific device or (2) set of devices, if the server uses a  
348 single queue to load balance between several devices. Each job set is disjoint; no job  
349 SHALL be represented in more than one MIB job set.

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

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

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

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

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

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

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

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

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

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

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

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

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

417 **2.1 System Configurations for the Job Monitoring MIB**

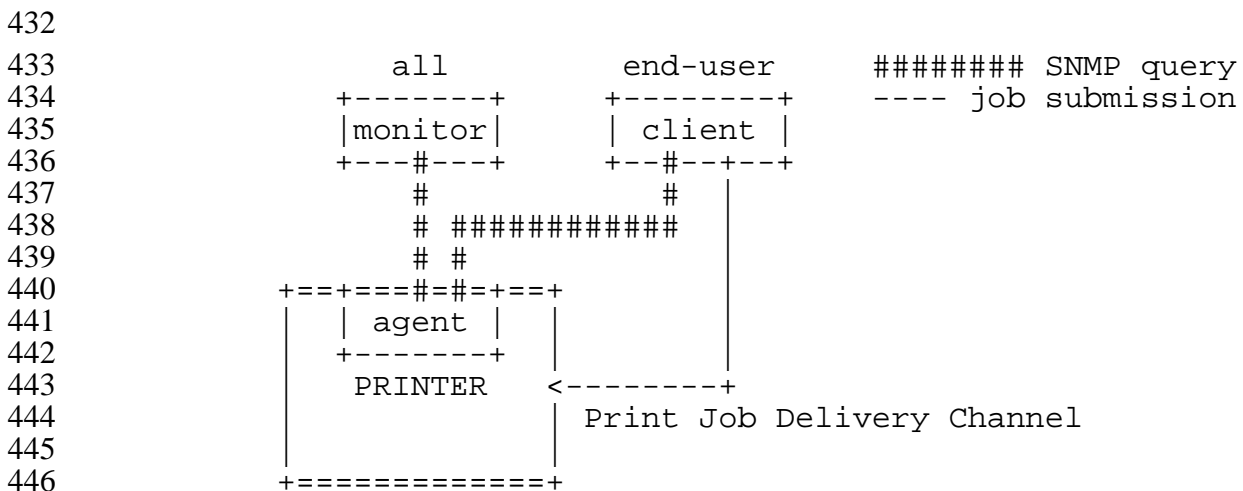
418 This section enumerates the three configurations in which the Job Monitoring MIB is  
 419 intended to be used. To simplify the pictures, the *devices* are shown as *printers*. See  
 420 ~~Goals~~ section 1.1 entitled "Types of Information in the MIB".

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

424 **2.1.1 Configuration 1 - client-printer**

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

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



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

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

- 450 1. Multiple **clients** MAY submit jobs to a **printer**.
- 451 2. Multiple **clients** MAY monitor a **printer**.

- 452 3. Multiple **monitors** MAY monitor a **printer**.
- 453 4. A **client** MAY submit jobs to multiple **printers**.
- 454 5. A **monitor** MAY monitor multiple **printers**.

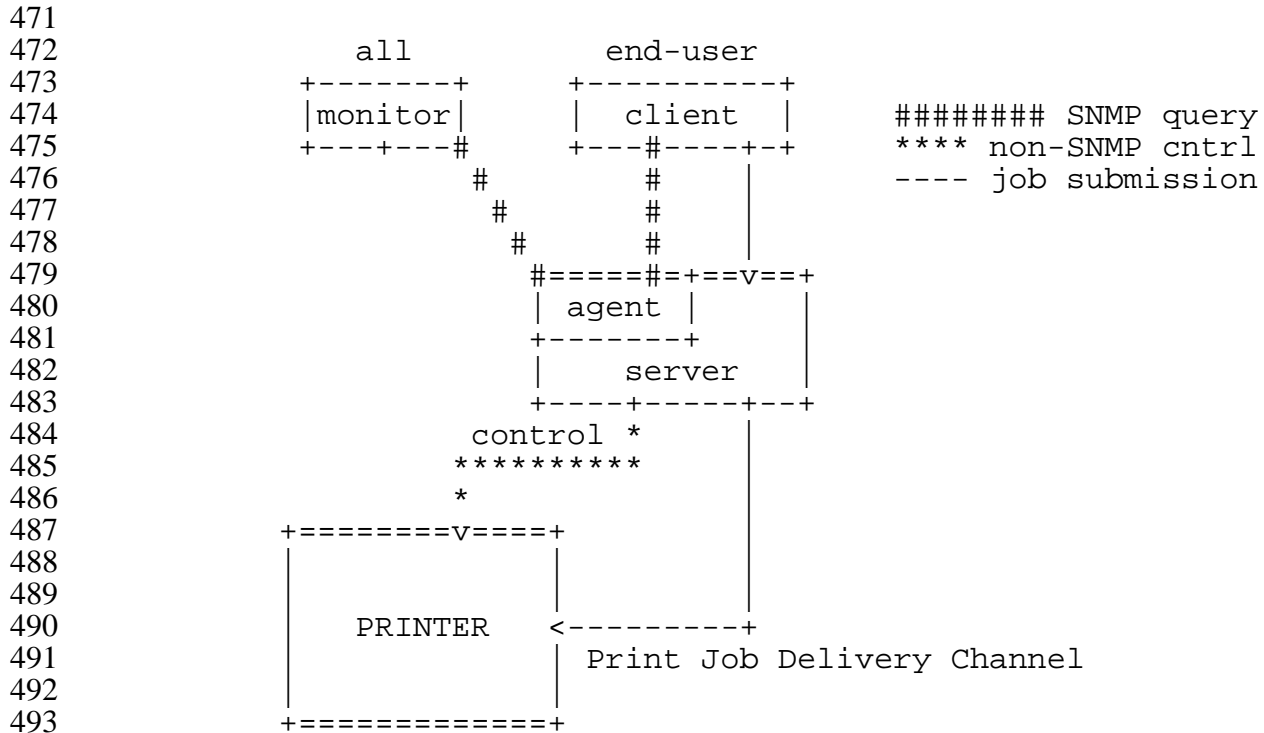
455 **2.1.2 Configuration 2 - client-server-printer - agent in the server**

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

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

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

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



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

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

- 497 1. Multiple **clients** MAY submit jobs to a **server**.
- 498 2. Multiple **clients** MAY monitor a **server**.
- 499 3. Multiple **monitors** MAY monitor a **server**.
- 500 4. A **client** MAY submit jobs to multiple **servers**.
- 501 5. A **monitor** MAY monitor multiple **servers**.
- 502 6. Multiple **servers** MAY submit jobs to a **printer**.
- 503 7. Multiple **servers** MAY control a **printer**.

504 **2.1.3 Configuration 3 - client-server-printer - client monitors printer agent and**  
505 **server**

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

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

- 511 1. The **server** using some undefined protocol to monitor jobs in the server (that  
512 does not contain the Job Monitoring MIB) AND





- 557 4. A **client** MAY submit jobs to multiple **servers**.  
558 5. A **monitor** MAY monitor multiple **servers**.  
559 6. Multiple **servers** MAY submit jobs to a **printer**.  
560 7. Multiple **servers** MAY control a **printer**.

### 561 3. Managed Object Usage

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

#### 563 3.1 Conformance Considerations

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

##### 567 3.1.1 Conformance Terminology

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

- 570 • "SHALL": indicates an action that the subject of the sentence must implement in  
571 order to claim conformance to this specification
- 572 • "MAY": indicates an action that the subject of the sentence does not have to  
573 implement in order to claim conformance to this specification, in other words that  
574 action is an implementation option
- 575 • "NEED NOT": indicates an action that the subject of the sentence does not have to  
576 implement in order to claim conformance to this specification. The verb "NEED  
577 NOT" is used instead of "may not", since "may not" sounds like a prohibition.
- 578 • "SHOULD": indicates an action that is recommended for the subject of the  
579 sentence to implement, but is not required, in order to claim conformance to this  
580 specification.

##### 581 3.1.2 Agent Conformance Requirements

582 A conforming agent:

- 583 1. SHALL implement *all* MANDATORY groups in this specification.  
584 2. SHALL implement any attributes if (1) the server or device supports the  
585 functionality represented by the attribute and (2) the information is available to  
586 the agent.  
587 3. SHOULD implement both forms of an attribute if it implements an attribute  
588 that permits a choice of INTEGER and OCTET STRING forms, since

589 implementing both forms may help management applications by giving them a  
590 choice of representations, since the representation are equivalent. See the  
591 **JmAttributeTypeTC** textual-convention.

592 NOTE - This MIB, like the Printer MIB, is written following the subset of SMIV2 that  
593 can be supported by SMIV1 and SNMPV1 implementations.

#### 594 3.1.2.1 MIB II System Group objects

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

#### 597 3.1.2.2 MIB II Interface Group objects

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

#### 600 3.1.2.3 Printer MIB objects

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

### 607 3.1.3 Job Monitoring Application Conformance Requirements

608 A conforming job monitoring application:

- 609 1. SHALL accept the full syntactic range for all objects in all MANDATORY  
610 groups and all MANDATORY attributes that are required to be implemented  
611 by an agent according to Section 3.1.2 and SHALL either present them to the  
612 user or ignore them.
- 613 2. SHALL accept the full syntactic range for *all* attributes, including enum and  
614 bit values specified in this specification and additional ones that may be  
615 registered with IANA and SHALL either present them to the user or ignore  
616 them. In particular, a conforming job monitoring application SHALL not  
617 malfunction when receiving any standard or registered enum or bit values.  
618 See Section 3.6 entitled "IANA Considerations".
- 619 3. SHALL NOT fail when operating with agents that materialize attributes *after*  
620 the job has been submitted, as opposed to when the job is submitted.
- 621 4. SHALL, if it supports a time attribute, accept either form of the time attribute,  
622 since agents are free to implement either time form.

### 623 3.2 The Job Tables and the Oldest Active and Newest Active Indexes

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

- 626 1. **jmGeneralJobSetIndex** - which job set
- 627 2. **jmJobIndex** - which job in the job set

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

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

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

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

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

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

658 Each time a new job is accepted by the server or device that the agent is providing access  
659 to AND that job is to be 'active' (**pending**, **processing**, or **processingStopped**, but not  
660 **pendingHeld**), the agent SHALL copy the value of the job's **jmJobIndex** to the

661 **jmGeneralNewestActiveJobIndex** object. If the new job is to be 'inactive'  
662 (**pendingHeld** state), the agent SHALL not change the value of  
663 **jmGeneralNewestActiveJobIndex** object (though the agent SHALL assign the next  
664 incremental **jmJobIndex** value to the job).

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

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

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

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

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

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

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

### 695 3.3 The Attribute Mechanism

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

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

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

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

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

#### 721 3.3.1 Conformance of Attribute Implementation

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

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

### 730 3.3.2 Useful, 'Unknown', and 'Other' Values for Objects and Attributes

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

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

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

748 Since each attribute is represented by a row consisting of both the  
749 **jmAttributeValueAsInteger** and **jmAttributeValueAsOctets** MANDATORY objects,  
750 SNMP requires that the agent SHALL always create an attribute row with both objects  
751 specified. However, for most attributes the agent SHALL return a "useful" value for one  
752 of the objects and SHALL return the 'other' value for the other object. For integer only  
753 attributes, the agent SHALL always return a zero-length string value for the  
754 **jmAttributeValueAsOctets** object. For octet string only attributes, the agent SHALL  
755 always return a '**-1**' value for the **jmAttributeValueAsInteger** object.

### 756 3.3.3 Data Sub-types and Attribute Naming Conventions

757 Many attributes are sub-typed to give a more specific data type than **Integer32** or  
758 **OCTET STRING**. The data sub-type of each attribute is indicated on the first line(s) of  
759 the description. Some attributes have several different data sub-type representations.  
760 When an attribute has both an **Integer32** data sub-type and an **OCTET STRING** data  
761 sub-type, the attribute can be represented in a single row in the **jmAttributeTable**. In  
762 this case, the data sub-type name is not included as the last part of the name of the  
763 attribute, e.g., **documentFormat(38)** which is both an enum and/or a name. When the  
764 data sub-types cannot be represented by a single row in the **jmAttributeTable**, each such  
765 representation is considered a separate attribute and is assigned a separate name and enum  
766 value. For these attributes, the name of the data sub-type is the last part of the name of

767 the attribute: **Name**, **Index**, **DateAndTime**, **TimeStamp**, etc. For example,  
 768 **documentFormatIndex(37)** is an index.

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

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

### 772 3.3.4 Single-Value (Row) Versus Multi-Value (MULTI-ROW) Attributes

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

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

### 786 3.3.5 Requested Attributes

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



### 793 3.3.6 Consumption Attributes

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

### 800 3.3.7 Index Value Attributes

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

## 807 3.4 Job Identification

808 There are a number of attributes that permit a user, operator or system administrator to  
809 identify jobs of interest, such as **jobName**, **jobOriginatingHost**, etc. In addition, there is  
810 a **jmJob-Submission-ID** object that is a text string table index. Being a table index allows  
811 a monitoring application to quickly locate and identify a particular job of interest that was  
812 submitted from a particular client by the user invoking the monitoring application. The  
813 Job Monitoring MIB needs to provide for identification of the job at both sides of the job  
814 submission process. The primary identification point is the client side. The **jmJob**  
815 **Submission-ID** allows the monitoring application to identify the job of interest from all  
816 the jobs currently "known" by the server or device. The value of jmJob-Submission-ID  
817 can be assigned by either the client's local system or a downstream server or device. The  
818 point of assignment depends on the job submission protocol in use.

819 The server/device-side identifier, called the **jmJobIndex** object, SHALL be assigned by  
820 the SNMP Job Monitoring MIB agent when the server or device accepts the jobs from  
821 submitting clients. The **jmJobIndex** object allows the interested party to obtain all  
822 objects desired that relate to a particular this job. See Section 3.2, entitled 'The Job Tables  
823 and the Oldest Active and Newest Active Indexes' for the specification of how the agent  
824 shall assign the jmJobIndex values.

825 NOTE - For a number of job submission protocols the server/device assigns an integer job  
826 identifier when accepting a job so that the submitting client can reference the job in  
827 subsequent protocol operations (For example, see IPP [ipp]). For such implementations,

828 it is recommended that the value of the job identifier and the value of jmJobIndex be the  
829 same, so that

830 The MIB provides a mapping table that maps each **jmJob-Submission-ID**  
831 value(generated by the client) to the corresponding **jmJobIndex** value generated by the  
832 agent, so that an application can determine the correct value for the **jmJobIndex** value for  
833 the job of interest in a single Get operation, given the Job Submission ID. See the  
834 **jmJobIDGroup**.

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

### 838 3.5 Internationalization Considerations

839 This section describes the internationalization considerations included in this MIB.

#### 840 3.5.1 'JmUTF8StringTC' for text generated by the server or device

841 There are a few objects and attributes that are represented using the Universal Multiple-  
842 Octet Coded Character Set (UCS) [ISO-10646] encoded as an octet string using the UTF-  
843 8 [UTF-8] character encoding scheme. The 'JmUTF8StringTC' textual convention is  
844 used to indicate UTF-8 text strings. These objects and attributes are always supplied (if  
845 implemented) by the agent, not by the job submitting client:

- 846 1. jmGeneralJobSetName object
- 847 2. processingMessage(6) attribute
- 848 3. physicalDevice(32) (name value) attribute

849 The coded character set for representing these objects and attributes SHALL be UTF-8 as  
850 recommended by RFC 2130 [RFC 2130] and the "IETF Policy on Character Sets and  
851 Language" [char-set policy].

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

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

858 ~~For simplicity, this specification assumes that the clients, job monitoring applications,~~  
859 ~~servers, and devices are all running in the same locale, including locales that use two-octet~~  
860 ~~coded character sets, such as ISO-10646 (Unicode). Job monitoring applications are~~  
861 ~~expected to understand the coded character set of the client (and job), server, or device.~~  
862 ~~No special means is provided for the monitor to discover the coded character set used by~~

863 jobs or by the server or device. This specification does *not* contain an object that indicates  
864 what locale the server or device is running in, let alone contain an object to control what  
865 locale the agent is to use to represent coded character set objects.

### 866 **3.5.2 'JmJobStringTC' for text generated by the job submitter**

867 All of the objects and attributes represented by the 'JmJobStringTC' textual-convention  
868 are either (1) supplied in the job submission protocol by the client that submits the job to  
869 the server or device or (2) are defaulted by the server or device if the job submitting client  
870 does not supply values. The agent SHALL represent these objects and attributes in the  
871 MIB either (1) in the coded character set as they were submitted or (2) MAY convert the  
872 coded character set to another coded character set or encoding scheme. In any case, the  
873 resulting coded character set representation SHOULD be UTF-8 [UTF-8], but SHALL be  
874 one in which the code positions from 0 to 31 SHALL not be used, 32 to 127 SHALL be  
875 US-ASCII [US-ASCII], 127 SHALL be unused, and the remaining code positions 128 to  
876 255 SHALL represent single-byte or multi-byte graphic characters structured according to  
877 ISO 2022 [ISO 2022] or SHALL be unused.

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

883 Examples of coded character sets which meet this criteria for use as the value of the  
884 **jobCodedCharSet** job attribute are: US-ASCII [US-ASCII], ISO 8859-1 (Latin-1) [ISO  
885 8859-1], any ISO 8859-n, HP Roman8, IBM Code Page 850, Windows Default 8-bit set,  
886 UTF-8 [UTF-8], US-ASCII plus JIS X0208-1990 Japanese [JIS X0208], US-ASCII plus  
887 GB2312-1980 PRC Chinese [GB2312]. See the IANA registry of coded character sets  
888 [IANA charsets].

889 Examples of coded character sets which do not meet this criteria are: national 7-bit sets  
890 conforming to ISO 646 (except US-ASCII), EBCDIC, and ISO 10646 (Unicode) [ISO-  
891 10646]. In order to represent Unicode characters, the UTF-8 [UTF-8] encoding scheme  
892 SHALL be used which has been assigned the MIBenum value of '106' by IANA.

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

### 895 **3.5.3 'DateAndTime' for representing the date and time**

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

899 **3.6 IANA Considerations**

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

906 **3.6.1 IANA Registration of enums**

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

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

918 3.6.1.1 Type 1 enumerations

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

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

922 3.6.1.2 Type 2 enumerations

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

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

- 927 1. JmUTF8StringTC
- 928 2. JmJobStringTC
- 929 3. JmTimeStampTC
- 930 4. JmFinishingTC [same enum values as IPP "finishing" attribute]
- 931 5. JmPrintQualityTC [same enum values as IPP "print-quality" attribute]
- 932 6. JmTonerEconomyTC
- 933 7. JmMediumTypeTC
- 934 8. JmJobSubmissionTypeTC

- 935 9. JmJobStateTC [same enum values as IPP "job-state" attribute]  
936 10. JmAttributeTypeTC

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

#### 940 3.6.1.3 Type 3 enumeration

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

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

#### 945 3.6.2 IANA Registration of type 2 bit values

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

- 947 1. JmJobServiceTypesTC  
948 2. JmJobStateReasons1TC  
949 3. JmJobStateReasons2TC  
950 4. JmJobStateReasons3TC  
951 5. JmJobStateReasons4TC

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

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

#### 957 3.6.3 IANA Registration of Job Submission Id Formats

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

#### 962 3.6.4 IANA Registration of MIME types/sub-types for document-formats

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

967 **3.7 Security Considerations**968 **3.7.1 Read-Write objects**

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

974 **3.7.2 Read-Only Objects In Other User's Jobs**

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

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

986 **3.8 Notifications**

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

991 **4. MIB specification**

992 The following pages constitute the actual Job Monitoring MIB.

```

993 Job-Monitoring-MIB DEFINITIONS ::= BEGIN
994
995 IMPORTS
    MODULE-IDENTITY, OBJECT-TYPE, experimental, Integer32
    TEXTUAL-CONVENTION
    MODULE-COMPLIANCE, OBJECT-GROUP
    FROM SNMPv2-SMI
    FROM SNMPv2-TC
    FROM SNMPv2-CONF;

    -- The following textual-conventions are needed
    -- to implement certain attributes, but are not
    -- needed to compile this MIB. They are
    -- provided here for convenience:
    -- hrDeviceIndex
    FROM HOST-RESOURCES-MIB
    -- DateAndTime
    FROM SNMPv2-TC
    -- PrtInterpreterLangFamilyTC
    -- CodedCharSet
    FROM Printer-MIB

996 -- Use the experimental (54) OID assigned to the Printer MIB[print-mib]
997 -- before it was published as RFC 1759.
998 -- Upon publication of the Job Monitoring MIB as an RFC, delete this
1000 -- comment and the line following this comment and change the
1001 -- reference of { temp 105 } (below) to { mib-2 X }.
1002 -- This will result in changing:
1003 -- 1 3 6 1 3 54 jobmonMIB(105)  to:
1004 -- 1 3 6 1 2 1 jobmonMIB(X)
1005 -- This will make it easier to translate prototypes to
1006 -- the standard namespace because the lengths of the OIDs won't
1007 -- change.
1008 temp OBJECT IDENTIFIER ::= { experimental 54 }
1009
1010 jobmonMIB MODULE-IDENTITY
1011     LAST-UPDATED "97080807240000Z"
1012     ORGANIZATION "IETF Printer MIB Working Group"
1013     CONTACT-INFO
1014         "Tom Hastings
1015         Postal: Xerox Corp.
1016         Mail stop ESAE-231
1017         701 S. Aviation Blvd.
1018         El Segundo, CA 90245
1019
1020         Tel: (301)333-6413
1021         Fax: (301)333-5514
1022         E-mail: hstings@cp10.es.xerox.com
1023
1024         Send comments to the printmib WG using the Job Monitoring
1025         Project (JMP) Mailing List: jmp@pwg.org
1026
1027         To learn how to subscribe to the JMP mailing list,
1028         send email to: jmp-request@pwg.org
1029

```

1030 For further information, access the PWG web page under 'JMP':  
 1031 <http://www.pwg.org/>"

1032 DESCRIPTION  
 1033 "The MIB module for monitoring job in servers, printers, and other devices.  
 1034  
 1035 File: draft-ietf-printmib-job-monitor-054.txt  
 1036 Version: 0.854"  
 1037 ::= { temp 105 }  
 1038  
 1039  
 1040  
 1041 -- Textual conventions for this MIB module  
 1042  
 1043  
 1044  
 1045 **JmUTF8StringTC ::= TEXTUAL-CONVENTION**  
 1046 DISPLAY-HINT "255a"  
 1047 STATUS current  
 1048 DESCRIPTION  
 1049 "To facilitate internationalization, this TC represents information taken from the ISO/IEC IS  
 1050 10646-1 character set, encoded as an octet string using the UTF-8 character encoding scheme.  
 1051  
 1052 NOTE - The values of objects and attributes using this textual convention are generated by the  
 1053 server or the device, not by the job submitter."  
 1054 REFERENCE  
 1055 "See section 3.5.1, 'JmUTF8StringTC' for text generated by the server or device'."  
 1056 SYNTAX OCTET STRING (SIZE (0..63))  
 1057  
 1058  
 1059  
 1060  
 1061 **JmJobStringTC ::= TEXTUAL-CONVENTION**  
 1062 STATUS current  
 1063 DESCRIPTION  
 1064 "To facilitate internationalization, this TC represents information using any coded character set  
 1065 registered by IANA that has the following properties: (1) code positions from 0 to 31 SHALL  
 1066 not be used, (2) 32 to 127 SHALL be US-ASCII [US-ASCII], (3) 127 SHALL be unused, and  
 1067 (4) the remaining code positions 128 to 255 SHALL represent single-byte or multi-byte graphic  
 1068 characters structured according to ISO 2022 [ISO 2022] or SHALL be unused. While it is  
 1069 recommended that the coded character set be UTF-8 [UTF-8], the actual coded character set  
 1070 SHALL be indicated by the value of the **jobCodedCharSet(7)** attribute for the job.  
 1071  
 1072 NOTE - The values of objects and attributes using this textual convention are either generated  
 1073 by the job submitter or defaulted by the server or device when the job submitter does not supply  
 1074 values."  
 1075 REFERENCE  
 1076 "See section 3.5.2, 'JmJobStringTC' for text generated by the job submitter'."



1077 SYNTAX OCTET STRING (SIZE (0..63))

1078

1079

1080

1081

1082 **JmTimeStampTC** ::= TEXTUAL-CONVENTION

1083 STATUS current

1084 DESCRIPTION

1085 "The simple time at which an event took place. The units SHALL be in seconds since the  
1086 system was booted.

1087  
1088 NOTE - **JmTimeStampTC** is defined in units of seconds, rather than 100ths of seconds, so as  
1089 to be simpler for agents to implement (even if they have to implement the 100ths of a second to  
1090 comply with implementing **sysUpTime** in MIB-II[mib-II].)

1091  
1092 NOTE - **JmTimeStampTC** is defined as an **Integer32** so that it can be used as a value of an  
1093 attribute, i.e., as a value of the **jmAttributeValueAsInteger** object. The **TimeStamp** textual-  
1094 convention defined in SMN Pv2-TC is defined as an **APPLICATION 3 IMPLICIT INTEGER**  
1095 tag, not an **Integer32**, so cannot be used in this MIB as one of the values of  
1096 **jmAttributeValueAsInteger**."

1097 SYNTAX **INTEGER(0..2147483647)**

1098

1099

1100

1101

1102 **JmJobSourcePlatformTypeTC** ::= TEXTUAL-CONVENTION

1103 STATUS current

1104 DESCRIPTION

1105 "The source platform type that can submit jobs to servers or devices in any of the 3  
1106 configurations."

1107 REFERENCE

1108 "This is a type 2 enumeration. See Section 3.6.1.2."

1109 SYNTAX **INTEGER {**

**other(1),**

**unknown(2),**

**sptUNIX(3),**

-- UNIX(tm)

**sptOS2(4),**

-- OS/2

**sptPCDOS(5),**

-- DOS

**sptNT(6),**

-- NT

**sptMVS(7),**

-- MVS

**sptVM(8),**

-- VM

**sptOS400(9),**

-- OS/400

**sptVMS(10),**

-- VMS

**sptWindows95(11),**

-- Windows95

**sptNetWare(33)**

-- NetWare

1110 }

1111

1111

1112  
 1113  
 1114  
 1115  
 1116 **JmFinishingTC** ::= TEXTUAL-CONVENTION  
 1117     STATUS     current  
 1118     DESCRIPTION  
 1119         "The type of finishing operation.  
 1120  
 1121         These values are the same as the enum values of the IPP 'finishings' attribute. See Section  
 1122         3.6.1.2.  
 1123  
 1124         **other(1)**,  
 1125             Some other finishing operation besides one of the specified or registered values.  
 1126  
 1127         **unknown(2)**,  
 1128             The finishing is unknown.  
 1129  
 1130         **none(3)**,  
 1131             Perform no finishing.  
 1132  
 1133         **staple(4)**,  
 1134             Bind the document(s) with one or more staples. The exact number and placement of the  
 1135             staples is site-defined.  
 1136  
 1137         **stapleTopLeft(5)**,  
 1138             Place one or more staples on the top left corner of the document(s).  
 1139  
 1140         **stapleBottomLeft(6)**,  
 1141             Place one or more staples on the bottom left corner of the document(s).  
 1142  
 1143         **stapleTopRight(7)**,  
 1144             Place one or more staples on the top right corner of the document(s).  
 1145  
 1146         **stapleBottomRight(8)**,  
 1147             Place one or more staples on the bottom right corner of the document(s).  
 1148  
 1149         **saddleStitch(9)**,  
 1150             Bind the document(s) with one or more staples (wire stitches) along the middle fold. The  
 1151             exact number and placement of the stitches is site-defined.  
 1152  
 1153         **edgeStitch(10)**,  
 1154             Bind the document(s) with one or more staples (wire stitches) along one edge. The exact  
 1155             number and placement of the staples is site-defined.  
 1156  
 1157         **punch(11)**,  
 1158             This value indicates that holes are required in the finished document. The exact number  
 1159             and placement of the holes is site-defined. The punch specification MAY be satisfied (in a

1160 site- and implementation-specific manner) either by drilling/punching, or by substituting  
 1161 pre-drilled media.  
 1162  
 1163 **cover(12),**  
 1164 This value is specified when it is desired to select a non-printed (or pre-printed) cover for  
 1165 the document. This does not supplant the specification of a printed cover (on cover stock  
 1166 medium) by the document itself.  
 1167  
 1168 **bind(13)**  
 1169 This value indicates that a binding is to be applied to the document; the type and  
 1170 placement of the binding is product-specific."  
 1171 REFERENCE  
 1172 "This is a type 2 enumeration. See Section 3.6.1.2."  
 1173 SYNTAX INTEGER {  
 1174 other(1),  
 1175 unknown(2),  
 1176 none(3),  
 1177 staple(4),  
 1178 stapleTopLeft(5),  
 1179 stapleBottomLeft(6),  
 1180 stapleTopRight(7),  
 1181 stapleBottomRight(8),  
 1182 saddleStitch(9),  
 1183 edgeStitch(10),  
 1184 punch(11),  
 1185 cover(12),  
 1186 bind(13)  
 1187 }  
 1188  
 1189  
 1190  
 1191  
 1192  
 1193 **JmPrintQualityTC ::= TEXTUAL-CONVENTION**  
 1194 STATUS current  
 1195 DESCRIPTION  
 1196 "Print quality settings."  
 1197  
 1198 These values are the same as the enum values of the IPP 'print-quality' attribute. See Section  
 1199 3.6.1.2."  
 1200 REFERENCE  
 1201 "This is a type 2 enumeration. See Section 3.6.1.2."  
 1202 SYNTAX INTEGER {  
 other(1), -- Not one of the specified or registered values.  
 --  
 unknown(2), -- The actual value is unknown.  
 draft(3), -- Lowest quality available on the printer.  
 normal(4), -- Normal or intermediate quality on the printer.

```

1203     }
1204
1205
1206
1207
1208 JmPrinterResolutionTC ::= TEXTUAL-CONVENTION
1209     STATUS      current
1210     DESCRIPTION
1211         "Printer resolutions.
1212
1213         Nine octets consisting of two 4-octet SIGNED-INTEGERS followed by a SIGNED-BYTE.
1214         The values are the same as those specified in the Printer MIB [printmib]. The first SIGNED-
1215         INTEGER contains the value of prtMarkerAddressabilityXFeedDir. The second SIGNED-
1216         INTEGER contains the value of prtMarkerAddressabilityFeedDir. The SIGNED-BYTE
1217         contains the value of prtMarkerAddressabilityUnit.
1218
1219         Note: the latter value is either 3 (tenThousandsOfInches) or 4 (micrometers) and the
1220         addressability is in 10,000 units of measure. Thus the SIGNED-INTEGERS represent integral
1221         values in either dots-per-inch or dots-per-centimeter.
1222
1223         The syntax is the same as the IPP 'printer-resolution' attribute. See Section 3.6.1.2."
1224     SYNTAX      OCTET STRING (SIZE(9))
1225
1226
1227
1228
1229
1230 JmTonerEconomyTC ::= TEXTUAL-CONVENTION
1231     STATUS      current
1232     DESCRIPTION
1233         "Toner economy settings."
1234     REFERENCE
1235         "This is a type 2 enumeration. See Section 3.6.1.2."
1236     SYNTAX      INTEGER {
1237         unknown(2),      -- unknown.
1238         off(3),         -- Off. Normal. Use full toner.
1239         on(4)           -- On. Use less toner than normal.
1240     }
1241
1242
1243 JmBooleanTC ::= TEXTUAL-CONVENTION
1244     STATUS      current

```

```

1245 DESCRIPTION
1246     "Boolean true or false value."
1247 REFERENCE
1248     "This is a type 2 enumeration. See Section 3.6.1.2."
1249 SYNTAX  INTEGER {
        unknown(2),      -- unknown.
        false(3),       -- FALSE.
        true(4)         -- TRUE.
1250     }
1251
1252
1253
1254
1255
1256 JmMediumTypeTC ::= TEXTUAL-CONVENTION
1257     STATUS current
1258     DESCRIPTION
1259         "Identifies the type of medium."
1260
1261     other(1),
1262         The type is neither one of the values listed in this specification nor a registered value.
1263
1264     unknown(2),
1265         The type is not known.
1266
1267     stationery(3),
1268         Separately cut sheets of an opaque material.
1269
1270     transparency(4),
1271         Separately cut sheets of a transparent material.
1272
1273     envelope(5),
1274         Envelopes that can be used for conventional mailing purposes.
1275
1276     envelopePlain(6),
1277         Envelopes that are not preprinted and have no windows.
1278
1279     envelopeWindow(7),
1280         Envelopes that have windows for addressing purposes.
1281
1282     continuousLong(8),
1283         Continuously connected sheets of an opaque material connected along the long edge.
1284
1285     continuousShort(9),
1286         Continuously connected sheets of an opaque material connected along the short edge.
1287
1288     tabStock(10),
1289         Media with tabs.

```

1290  
 1291 **multiPartForm(11),**  
 1292 Form medium composed of multiple layers not pre-attached to one another; each sheet  
 1293 MAY be drawn separately from an input source.  
 1294  
 1295 **labels(12),**  
 1296 Label-stock.  
 1297  
 1298 **multiLayer(13)**  
 1299 Form medium composed of multiple layers which are pre-attached to one another, e.g. for  
 1300 use with impact printers."  
 1301 REFERENCE  
 1302 "This is a type 2 enumeration. See Section 3.6.1.2."  
 1303 SYNTAX INTEGER {  
 1304 other(1),  
 1305 unknown(2),  
 1306 stationery(3),  
 1307 transparency(4),  
 1308 envelope(5),  
 1309 envelopePlain(6),  
 1310 envelopeWindow(7),  
 1311 continuousLong(8),  
 1312 continuousShort(9),  
 1313 tabStock(10),  
 1314 multiPartForm(11),  
 1315 labels(12),  
 1316 multiLayer(13)  
 1317 }  
 1318  
 1319  
 1320  
 1321  
 1322  
 1323 **JmJobSubmissionTypeTC ::= TEXTUAL-CONVENTION**  
 1324 STATUS current  
 1325 DESCRIPTION  
 1326 "Identifies the format type of a job submission ID.  
 1327  
 1328 The ASCII characters '0-9', 'A-Z', and 'a-z' are assigned in order giving 62 possible formats.  
 1329  
 1330 Each job submission ID is a fixed-length, 48-octet printable ASCII coded character string,  
 1331 consisting of the following fields:  
 1332  
 1333 octet 1 The format letter.  
 1334 octets 2-40 A 39-character, ASCII trailing SPACE filled  
 1335 field specified by the format letter, if the  
 1336 data is less than 39 ASCII characters.  
 1337 octets 41-48 A sequential or random number to make the ID

1338  
1339  
1340  
1341  
1342  
1343  
1344  
1345  
1346  
1347  
1348  
1349  
1350  
1351  
1352  
1353  
1354  
1355  
1356  
1357  
1358  
1359  
1360  
1361  
1362  
1363  
1364  
1365  
1366  
1367  
1368  
1369  
1370  
1371  
1372  
1373  
1374  
1375  
1376  
1377  
1378  
1379  
1380  
1381  
1382  
1383  
1384  
1385  
1386

quasi-unique.

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

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

Format Letter	Description
'0'	octets 2-40: last 39 bytes of the <b>jmJobOwner</b> object. octets 41-48: 8-decimal-digit sequential number This format is reserved to agents for use when the client does not supply a job submission ID. Clients wishing to use a job submission ID that incorporates the job owner, SHALL use format '8', not format '0', in order to reduce the chances of one client assigning the same ID as the agent when receiving a job from another client that does not supply a job submission id.

NOTE - other formats may be registered that are reserved to the agent for use when the client does not supply a job submission ID.

'1'	octets 2-40: last 39 bytes of the <b>jobName</b> attribute. octets 41-48: 8-decimal-digit random number
'2'	octets 2-40: Client MAC address: in hexadecimal with each nibble of the 6 octet address being '0'-'9' or 'A' - 'F' (uppercase only). Most significant octet first. octets 41-48: 8-decimal-digit sequential number
'3'	octets 2-40: last 39 bytes of the client URL [URI-spec]. octets 41-48: 8-decimal-digit sequential number
'4'	octets 2-40: last 39 bytes of the URI [URI-spec] assigned by the server or device to the job when the job was submitted for processing. octets 41-48: 8-decimal-digit sequential number
'5'	octets 2-40: last 39 bytes of a user number, such as POSIX user number. octets 41-48: 8-decimal-digit sequential number

- 1387 '6' octets 2-40: last 39 bytes of the user account
- 1388 number.
- 1389 octets 41-48: 8-decimal-digit sequential number
- 1390
- 1391 '7' octets 2-40: last 39 bytes of the DTMF incoming
- 1392 FAX routing number.
- 1393 octets 41-48: 8-decimal-digit sequential number
- 1394
- 1395 '8' octets 2-40: last 39 bytes of the job owner name
- 1396 (that the agent returns in the **jmJobOwner** object).
- 1397 octets 41-48: 8-decimal-digit sequential number
- 1398

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

REFERENCE

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

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

**JmJobStateTC ::= TEXTUAL-CONVENTION**

STATUS current

DESCRIPTION

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

The following figure shows the normal job state transitions:

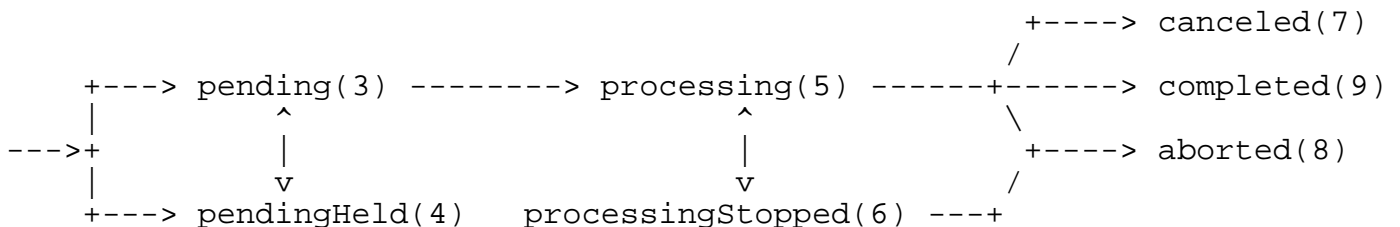


Figure 4 - Normal Job State Transitions



1434  
 1435 Normally a job progresses from left to right. Other state transitions are unlikely, but are not  
 1436 forbidden. Not shown are the transitions to the **canceled** state from the **pending**,  
 1437 **pendingHeld**, **processing**, and **processingStopped** states.  
 1438  
 1439 Jobs in the **pending**, **processing**, and **processingStopped** states are called 'active', while jobs in  
 1440 the **pendingHeld**, **canceled**, **aborted**, and **completed** are called 'inactive'.  
 1441  
 1442 These values are the same as the enum values of the IPP 'job-state' job attribute. See Section  
 1443 3.6.1.2.  
 1444  
 1445 **other(1),**  
 1446     The job state is *not* one of the defined states.  
 1447  
 1448 **unknown(2),**  
 1449     The job state is *not* known, or its state is indeterminate.  
 1450  
 1451 **pending(3),**  
 1452     The job is a candidate to start processing, but is not yet processing.  
 1453  
 1454 **pendingHeld(4),**  
 1455     The job is not a candidate for processing for any number of reasons but will return to the  
 1456 pending state as soon as the reasons are no longer present. The job's  
 1457 **jmJobStateReasons1** object and/or **jobStateReasonsN** ( $N=2..4$ ) attributes SHALL  
 1458 indicate why the job is no longer a candidate for processing. The reasons are represented  
 1459 as bits in the **jmJobStateReasons1** object and/or **jobStateReasonsN** ( $N=2..4$ ) attributes.  
 1460 See the **JmJobStateReasonsNTC** ( $N=1..4$ ) textual convention for the specification of  
 1461 each reason.  
 1462  
 1463 **processing(5),**  
 1464     Either:  
 1465  
 1466         1. The job is using, or is attempting to use, one or more document transforms which  
 1467 include (1) purely software processes that are interpreting a PDL, and (2) hardware  
 1468 devices that are interpreting a PDL, making marks on a medium, and/or performing  
 1469 finishing, such as stapling, etc.  
 1470  
 1471         OR  
 1472  
 1473         2. (configuration 2) the server has made the job ready for printing, but the output device is  
 1474 not yet printing it, either because the job hasn't reached the output device or because the  
 1475 job is queued in the output device or some other spooler, awaiting the output device to  
 1476 print it.  
 1477  
 1478     When the job is in the **processing** state, the entire job state includes the detailed status  
 1479 represented in the device MIB indicated by the **hrDeviceIndex** value of the job's  
 1480 **physicalDevice** attribute, if the agent implements such a device MIB.  
 1481

1482 Implementations MAY, though they NEED NOT, include additional values in the job's  
 1483 **jmJobStateReasons1** object to indicate the progress of the job, such as adding the  
 1484 **jobPrinting** value to indicate when the device is actually making marks on a medium.  
 1485

1486 **processingStopped(6),**  
 1487 The job has stopped while processing for any number of reasons and will return to the  
 1488 **processing** state as soon as the reasons are no longer present.  
 1489

1490 The job's **jmJobStateReasons1** object and/or the job's **jobStateReasonsN** ( $N=2..4$ )  
 1491 attributes MAY indicate why the job has stopped processing. For example, if the output  
 1492 device is stopped, the **deviceStopped** value MAY be included in the job's  
 1493 **jmJobStateReasons1** object.  
 1494

1495 NOTE - When an output device is stopped, the device usually indicates its condition in  
 1496 human readable form at the device. The management application can obtain more  
 1497 complete device status remotely by querying the appropriate device MIB using the job's  
 1498 **deviceIndex** attribute(s), if the agent implements such a device MIB  
 1499

1500 **canceled(7),**  
 1501 A client has canceled the job and the job is either: (1) in the process of being terminated by  
 1502 the server or device or (2) has completed terminating. The job's **jmJobStateReasons1**  
 1503 object SHOULD contain either the **canceledByUser** or **canceledByOperator** value.  
 1504

1505 **aborted(8),**  
 1506 The job has been aborted by the system, usually while the job was in the processing or  
 1507 **processingStopped** state.  
 1508

1509 **completed(9)**  
 1510 The job has completed successfully or with warnings or errors after processing and all of  
 1511 the media have been successfully stacked in the appropriate output bin(s). The job's  
 1512 **jmJobStateReasons1** object SHOULD contain one of: **completedSuccessfully**,  
 1513 **completedWithWarnings**, or **completedWithErrors** values."  
 1514 REFERENCE  
 1515 "This is a type 2 enumeration. See Section 3.6.1.2."  
 1516 SYNTAX INTEGER {  
 1517 **other(1)**,  
 1518 **unknown(2)**,  
 1519 **pending(3)**,  
 1520 **pendingHeld(4)**,  
 1521 **processing(5)**,  
 1522 **processingStopped(6)**,  
 1523 **canceled(7)**,  
 1524 **aborted(8)**,  
 1525 **completed(9)**  
 1526 }  
 1527

1528

1529 **JmAttributeTypeTC ::= TEXTUAL-CONVENTION**

1530  
1531  
1532  
1533  
1534  
1535  
1536  
1537  
1538  
1539  
1540  
1541  
1542  
1543  
1544  
1545  
1546  
1547  
1548  
1549  
1550  
1551  
1552  
1553  
1554  
1555  
1556  
1557  
1558  
1559  
1560  
1561  
1562  
1563  
1564  
1565  
1566  
1567  
1568  
1569  
1570  
1571  
1572  
1573  
1574  
1575  
1576  
1577  
1578

STATUS current  
DESCRIPTION

"The type of the attribute which identifies the attribute.

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

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

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

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

NOTE: No attribute name exceeds 31 characters.

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

<b>jmAttributeTypeIndex</b> -----	<b>Datatype</b> -----
<b>other(1),</b>	<b>Integer32(-2..2147483647)</b> <b>AND/OR</b> <b>OCTET STRING(SIZE(0..63))</b>
INTEGER: and/or OCTETS: An attribute that is not in the list and/or that has not been approved and registered with IANA.	

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

**jobStateReasons2(3),** **JmJobStateReasons2TC**  
INTEGER: Additional information about the job's current state that augments the **jmJobState** object. See the description under the **JmJobStateReasons1TC** textual-convention.

**jobStateReasons3(4),** **JmJobStateReasons3TC**  
INTEGER: Additional information about the job's current state that augments the

1579 **jmJobState** object. See the description under **JmJobStateReasons1TC** textual-  
1580 convention.

1581  
1582 **jobStateReasons4(5),** **JmJobStateReasons4TC**  
1583 INTEGER: Additional information about the job's current state that augments the  
1584 **jmJobState** object. See the description under **JmJobStateReasons1TC** textual-  
1585 convention.

1586  
1587 **processingMessage(6),** **JmUTF8StringTCOCTET**  
1588 **STRING(SIZE(0..63))**  
1589 OCTETS: MULTI-ROW: A coded character set message that is generated by the server  
1590 or device during the processing of the job as a simple form of processing log to show  
1591 progress and any problems.

1592  
1593 There is no restriction for the same message occurring in multiple rows.

1594  
1595 **jobCodedCharSet(7),** **CodedCharSet**  
1596 INTEGER: The MIBenum identifier of the coded character set that the agent is using to  
1597 represent coded character set objects and attributes of type 'JmJobStringTC'. These  
1598 coded character set objects and attributes are either: (1) supplied by the job submitting  
1599 client or (2) defaulted by the server or device when omitted by the job submitting client.  
1600 The agent SHALL represent these objects and attributes in the MIB either (1) in the coded  
1601 character set as they were submitted or (2) MAY convert the coded character set to  
1602 another coded character set or encoding scheme as identified by the **jobCodedCharSet**  
1603 attribute.

1604  
1605 These MIBenum values are assigned by IANA [IANA-charsets] when the coded character  
1606 sets are registered. The coded character set SHALL be one of the ones registered with  
1607 IANA [IANA] and the enum value uses the **CodedCharSet** textual-convention from the  
1608 Printer MIB. See the **JmJobStringTC** textual-convention.

1609  
1610 If the agent does not know what coded character set was used by the job submitting client,  
1611 the agent SHALL return the 'unknown(2)' value for the **jobCodedCharSet** attribute for  
1612 the job. See Section 3.5.2, entitled "JmJobStringTC' for text generated by the job  
1613 submitter'.

1614  
1615  
1616  
1617 ++++++  
1618 + **Job Identification attributes**  
1619 +  
1620 + **The following attributes help an end user, a system**  
1621 **+ operator, or an accounting program identify a job.**  
1622 ++++++

1623  
1624  
1625  
1626 **jobAccountName(21),** **JmJobStringTCOCTET**  
1627 **STRING(SIZE(0..63))**

1628 OCTETS: Arbitrary binary information which MAY be coded character set data or  
 1629 encrypted data supplied by the submitting user for use by accounting services to allocate  
 1630 or categorize charges for services provided, such as a customer account name or number.  
 1631

1632 NOTE: This attribute NEED NOT be printable characters.  
 1633

1634 **serverAssignedJobName(22),** **JmJobStringTCOCTET**  
 1635 **STRING(SIZE(0..63))**

1636 OCTETS: Configuration 3 only: The human readable string name, number, or ID of the  
 1637 job as assigned by the server that submitted the job to the device that the agent is  
 1638 providing access to with this MIB.  
 1639

1640 NOTE - This attribute is intended for enabling a user to find his/her job that a server  
 1641 submitted to a device when either the client does not support the **jmJobSubmissionID** or  
 1642 the server does not pass the **jmJobSubmissionID** through to the device.  
 1643

1644 **jobName(23),** **JmJobStringTCOCTET**  
 1645 **STRING(SIZE(0..63))**

1646 OCTETS: The human readable string name of the job as assigned by the submitting user  
 1647 to help the user distinguish between his/her various jobs. This name does not need to be  
 1648 unique.  
 1649

1650 This attribute is intended for enabling a user or the user's application to convey a job name  
 1651 that MAY be printed on a start sheet, returned in a **query** result, or used in notification or  
 1652 logging messages.  
 1653

1654 In order to assist users to find their jobs for job submission protocols that don't supply a  
 1655 **jmJobSubmissionID**, the agent SHOULD maintain the **jobName** attribute for the time  
 1656 specified by the **jmGeneralJobPersistence** object, rather than the (shorter)  
 1657 **jmGeneralAttributePersistence** object.  
 1658

1659 If this attribute is not specified when the job is submitted, no job name is assumed, but  
 1660 implementation specific defaults are allowed, such as the value of the **documentName**  
 1661 attribute of the first document in the job or the **fileName** attribute of the first document in  
 1662 the job.  
 1663

1664 The **jobName** attribute is distinguished from the **jobComment** attribute, in that the  
 1665 **jobName** attribute is intended to permit the submitting user to distinguish between  
 1666 different jobs that he/she has submitted. The **jobComment** attribute is intended to be free  
 1667 form additional information that a user might wish to use to communicate with  
 1668 himself/herself, such as a reminder of what to do with the results or to indicate a different  
 1669 set of input parameters were tried in several different job submissions.  
 1670

1671 **jobServiceTypes(24),** **JmJobServiceTypesTC**

1672 **INTEGER:** Specifies the type(s) of service to which the job has been submitted (print,  
 1673 fax, scan, etc.). The service type is bit encoded with each job service type so that more  
 1674 general and arbitrary services can be created, such as services with more than one  
 1675 destination type, or ones with only a source or only a destination. For example, a job  
 1676 service might **scan**, **faxOut**, and **print** a single job. In this case, three bits would be set in

1677 the **jobServiceTypes** attribute, corresponding to the hexadecimal values: **0x8 + 0x20 +**  
 1678 **0x4**, respectively, yielding: **0x2C**.  
 1679  
 1680 Whether this attribute is set from a job attribute supplied by the job submission client or is  
 1681 set by the recipient job submission server or device depends on the job submission  
 1682 protocol. This attribute **SHALL** be implemented if the server or device has other types in  
 1683 addition to or instead of printing.  
 1684  
 1685 One of the purposes of this attribute is to permit a requester to filter out jobs that are not  
 1686 of interest. For example, a printer operator may only be interested in jobs that include  
 1687 printing.  
 1688  
 1689 **jobSourceChannelIndex(25),** Integer32(0..2147483647)  
 1690 INTEGER: The index of the row in the associated Printer MIB[print-mib] of the channel  
 1691 which is the source of the print job.  
 1692  
 1693 **jobSourcePlatformType(26),** **JmJobSourcePlatformTypeTC**  
 1694 INTEGER: The source platform type of the immediate upstream submitter that submitted  
 1695 the job to the server (configuration 2) or device (configuration 1 and 3) to which the agent  
 1696 is providing access. For configuration 1, this is the type of the client that submitted the  
 1697 job to the device; for configuration 2, this is the type of the client that submitted the job  
 1698 to the server; and for configuration 3, this is the type of the server that submitted the job  
 1699 to the device.  
 1700  
 1701 **submittingServerName(27),** **JmJobStringTCOCTET**  
 1702 **STRING(SIZE(0..63))**  
 1703 OCTETS: For configuration 3 only: The administrative name of the server that submitted  
 1704 the job to the device.  
 1705  
 1706 **submittingApplicationName(28),** **JmJobStringTCOCTET**  
 1707 **STRING(SIZE(0..63))**  
 1708 OCTETS: The name of the client application (not the server in configuration 3) that  
 1709 submitted the job to the server or device.  
 1710  
 1711 **jobOriginatingHost(29),** **JmJobStringTCOCTET**  
 1712 **STRING(SIZE(0..63))**  
 1713 OCTETS: The name of the client host (not the server host name in configuration 3) that  
 1714 submitted the job to the server or device.  
 1715  
 1716 **deviceNameRequested(30),** **JmJobStringTCOCTET**  
 1717 **STRING(SIZE(0..63))**  
 1718 OCTETS: The administratively defined coded character set name of the target device  
 1719 requested by the submitting user. For configuration 1, its value corresponds to the Printer  
 1720 MIB[print-mib]: **prtGeneralPrinterName** object. For configuration 2 and 3, its value is  
 1721 the name of the logical or physical device that the user supplied to indicate to the server  
 1722 on which device(s) they wanted the job to be processed.  
 1723  
 1724 **queueNameRequested(31),** **JmJobStringTCOCTET**  
 1725 **STRING(SIZE(0..63))**

1726 OCTETS: The administratively defined coded character set name of the target queue  
 1727 requested by the submitting user. For configuration 1, its value corresponds to the queue  
 1728 in the device for which the agent is providing access. For configuration 2 and 3, its value  
 1729 is the name of the queue that the user supplied to indicate to the server on which device(s)  
 1730 they wanted the job to be processed.  
 1731

1732 NOTE - typically an implementation SHOULD support either the **deviceNameRequested**  
 1733 or **queueNameRequested** attribute, but not both.  
 1734

1735 | **physicalDevice(32),** **hrDeviceIndex**  
 1736 | **AND/OR**  
 1737 | **JmUTF8StringTCOCTET**

1738 | **STRING(SIZE(0..63))**  
 1739 | INTEGER: MULTI-ROW: The index of the physical device MIB instance  
 1740 | requested/used, such as the Printer MIB[print-mib]. This value is an **hrDeviceIndex**  
 1741 | value. See the Host Resources MIB[hr-mib].  
 1742

1743 | AND/OR

1744 | OCTETS: MULTI-ROW: The name of the physical device to which the job is assigned.  
 1745

1746 | **numberOfDocuments(33),** **Integer32(-2..2147483647)**  
 1747 | INTEGER: The number of documents in this job.  
 1748

1749 | **fileName(34),** **JmJobStringTCOCTET**  
 1750 | **STRING(SIZE(0..63))**

1751 | OCTETS: MULTI-ROW: The coded character set file name or URI[URI-spec] of the  
 1752 | document.  
 1753

1754 | There is no restriction on the same file name occurring in multiple rows.  
 1755

1756 | **documentName(35),** **JmJobStringTCOCTET**  
 1757 | **STRING(SIZE(0..63))**

1758 | OCTETS: MULTI-ROW: The coded character set name of the document.  
 1759

1760 | There is no restriction on the same document name occurring in multiple rows.  
 1761

1762 | **jobComment(36),** **JmJobStringTCOCTET**  
 1763 | **STRING(SIZE(0..63))**

1764 | OCTETS: An arbitrary human-readable coded character text string supplied by the  
 1765 | submitting user or the job submitting application program for any purpose. For example,  
 1766 | a user might indicate what he/she is going to do with the printed output or the job  
 1767 | submitting application program might indicate how the document was produced.  
 1768

1769 | The **jobComment** attribute is not intended to be a name; see the **jobName** attribute.  
 1770

1771 | **documentFormatIndex(37),** **Integer32(0..2147483647)**

1772 | INTEGER: MULTI-ROW: The index in the **prtInterpreterTable** in the Printer  
 1773 | MIB[print-mib] of the page description language (PDL) or control language interpreter  
 1774

1775 that this job requires/uses. A document or a job MAY use more than one PDL or control  
1776 language.

1777  
1778 NOTE - As with all intensive attributes where multiple rows are allowed, there SHALL be  
1779 only one distinct row for each distinct interpreter; there SHALL be no duplicates.

1780  
1781 NOTE - This attribute type is intended to be used with an agent that implements the  
1782 Printer MIB and SHALL not be used if the agent does not implement the Printer MIB.  
1783 Such an agent SHALL use the **documentFormat** attribute instead.

1784  
1785 **documentFormat(38),** **PrtInterpreterLangFamilyTC**  
1786 **AND/OR**  
1787 **OCTET STRING(SIZE(0..63))**

1788 INTEGER: MULTI-ROW: The interpreter language family corresponding to the Printer  
1789 MIB[print-mib] **prtInterpreterLangFamily** object, that this job requires/uses. A  
1790 document or a job MAY use more than one PDL or control language.

1791  
1792 AND/OR

1793  
1794 OCTETS: MULTI-ROW: The document format registered as a media type[iana-media-  
1795 types], i.e., the name of the MIME content-type/subtype. Examples:  
1796 'application/postscript', 'application/vnd.hp-PCL', and 'application/pdf'

1797  
1798  
1799 ++++++  
1800 + **Job Parameter attributes**  
1801 +  
1802 + **The following attributes represent input parameters**  
1803 + **supplied by the submitting client in the job submission**  
1804 + **protocol.**

1805 ++++++  
1806  
1807 **jobPriority(50),** **Integer32(1..100)**

1808 INTEGER: The priority for scheduling the job. It is used by servers and devices that  
1809 employ a priority-based scheduling algorithm.

1810  
1811 A higher value specifies a higher priority. The value **1** is defined to indicate the lowest  
1812 possible priority (a job which a priority-based scheduling algorithm SHALL pass over in  
1813 favor of higher priority jobs). The value **100** is defined to indicate the highest possible  
1814 priority. Priority is expected to be evenly or 'normally' distributed across this range. The  
1815 mapping of vendor-defined priority over this range is implementation-specific.

1816  
1817 **jobProcessAfterDateAndTime(51),** **DateAndTime (SNMPv2-TC)**

1818 OCTETS: The calendar date and time of day after which the job SHALL become a  
1819 candidate to be scheduled for processing. If the value of this attribute is in the future, the  
1820 server SHALL set the value of the job's **jmJobState** object to **pendingHeld** and add the  
1821 **jobProcessAfterSpecified** bit value to the job's **jmJobStateReasons1** object. When the  
1822 specified date and time arrives, the server SHALL remove the **jobProcessAfterSpecified**



1823 bit value from the job's **jmJobStateReasons1** object and, if no other reasons remain,  
1824 SHALL change the job's **jmJobState** object to **pending**.

1825  
1826 **jobHold(52),** **JmBooleanTC**  
1827 INTEGER: If the value is 'true(4)', a client has explicitly specified that the job is to be  
1828 held until explicitly released. Until the job is explicitly released by a client, the job SHALL  
1829 be in the **pendingHeld** state with the **jobHoldSpecified** value in the  
1830 **jmJobStateReasons1** attribute.

1831  
1832 **jobHoldUntil(53),** **JmJobStringTCOCTET**  
1833 **STRING(SIZE(0..63))**  
1834 OCTETS: The named time period during which the job SHALL become a candidate for  
1835 processing, such as 'evening', 'night', 'weekend', 'second-shift', 'third-shift', etc., as  
1836 defined by the system administrator. See IPP [ipp-model] for the standard keyword  
1837 values. Until that time period arrives, the job SHALL be in the **pendingHeld** state with  
1838 the **jobHoldUntilSpecified** value in the **jmJobStateReasons1** object. The value '**no-**  
1839 **hold**' SHALL indicate explicitly that no time period has been specified; the absence of this  
1840 attribute SHALL indicate implicitly that no time period has been specified.

1841  
1842 **outputBin(54),** **Integer32(0..2147483647)**  
1843 **AND/OR**  
1844 **JmJobStringTCOCTET**  
1845 **STRING(SIZE(0..63))**  
1846 INTEGER: MULTI-ROW: The output subunit index in the Printer MIB[print-mib]  
1847  
1848 **AND/OR**  
1849  
1850 OCTETS: the name or number (represented as ASCII digits) of the output bin to which  
1851 all or part of the job is placed in.

1852  
1853 **sides(55),** **Integer32(-2..2)**  
1854 INTEGER: MULTI-ROW: The number of sides, '1' or '2', that any document in this job  
1855 requires/used.

1856  
1857 **finishing(56),** **JmFinishingTC**  
1858 INTEGER: MULTI-ROW: Type of finishing that any document in this job requires/used.

1859  
1860  
1861 ++++++  
1862 + **Image Quality attributes (requested and consumed)**  
1863 +  
1864 + **For devices that can vary the image quality.**  
1865 ++++++

1866  
1867 **printQualityRequested(70),** **JmPrintQualityTC**  
1868 INTEGER: MULTI-ROW: The print quality selection requested for a document in the  
1869 job for printers that allow quality differentiation.

1870

1871 **printQualityUsed(71),** **JmPrintQualityTC**  
1872 INTEGER: MULTI-ROW: The print quality selection actually used by a document in the  
1873 job for printers that allow quality differentiation.  
1874

1875 **printerResolutionRequested(72),** **JmPrinterResolutionTC**  
1876 OCTETS: MULTI-ROW: The printer resolution requested for a document in the job for  
1877 printers that support resolution selection.  
1878

1879 **printerResolutionUsed(73),** **JmPrinterResolutionTC**  
1880 OCTETS: MULTI-ROW: The printer resolution actually used by a document in the job  
1881 for printers that support resolution selection.  
1882

1883 **tonerEcomonyRequested(74),** **JmTonerEcomonyTC**  
1884 INTEGER: MULTI-ROW: The toner economy~~print quality~~ selection requested for  
1885 documents in the job for printers that allow toner economy~~quality~~ differentiation.  
1886

1887 **tonerEcomonyUsed(75),** **JmTonerEcomonyTC**  
1888 INTEGER: MULTI-ROW: The toner economy~~print quality~~ selection actually used by  
1889 documents in the job for printers that allow toner economy~~quality~~ differentiation.  
1890

1891 **tonerDensityRequested(76),** **Integer32(-2..100)**  
1892 INTEGER: MULTI-ROW: The toner density requested for a document in this job for  
1893 devices that can vary toner density levels. Level 1 is the lowest density and level 100 is  
1894 the highest density level. Devices with a smaller range, SHALL map the 1-100 range  
1895 evenly onto the implemented range.  
1896

1897 **tonerDensityUsed(77),** **Integer32(-2..100)**  
1898 INTEGER: MULTI-ROW: The toner density used by documents in this job for devices  
1899 that can vary toner density levels. Level 1 is the lowest density and level 100 is the highest  
1900 density level. Devices with a smaller range, SHALL map the 1-100 range evenly onto the  
1901 implemented range.  
1902

1903  
1904 ++++++  
1905 + **Job Progress attributes (requested and consumed)**  
1906 +  
1907 + **Pairs of these attributes can be used by monitoring**  
1908 + **applications to show an indication of relative progress**  
1909 + **to users.**  
1910 ++++++

1911

1912 **jobCopiesRequested(90),** **Integer32(-2..2147483647)**  
1913 INTEGER: The number of copies of the entire job that are to be produced.  
1914

1915 **jobCopiesCompleted(91),** **Integer32(-2..2147483647)**  
1916 INTEGER: The number of copies of the entire job that have been completed so far.  
1917

1918 **documentCopiesRequested(92),** **Integer32(-2..2147483647)**  
1919 INTEGER: The total count of the number of document copies requested for the job as a

1920 | whole. If there are documents A, B, and C, and document B is specified to produce 4  
1921 | copies, the number of document copies requested is 6 for the job.

1922 |  
1923 | This attribute SHALL be used only when a job has multiple documents. The  
1924 | **jobCopiesRequested** attribute SHALL be used when the job has only one document.

1925 |  
1926 | **documentCopiesCompleted(93), Integer32(-2..2147483647)**  
1927 | INTEGER: The total count of the number of document copies completed so far for the  
1928 | job as a whole. If there are documents A, B, and C, and document B is specified to  
1929 | produce 4 copies, the number of document copies starts a 0 and runs up to 6 for the job as  
1930 | the job processes.

1931 |  
1932 | This attribute SHALL be used only when a job has multiple documents. The  
1933 | **jobCopiesCompleted** attribute SHALL be used when the job has only one document.

1934 |  
1935 | **jobKOctetsTransferred(94), Integer32(-2..2147483647)**  
1936 | INTEGER: The number of K (1024) octets transferred to the server or device to which  
1937 | the agent is providing access. This count is independent of the number of copies of the  
1938 | job or documents that will be produced, but it is only a measure of the number of bytes  
1939 | transferred to the server or device.

1940 |  
1941 | The agent SHALL round the actual number of octets transferred up to the next higher K.  
1942 | Thus 0 octets SHALL be represented as '0', 1-1024 octets SHALL BE represented as '1',  
1943 | 1025-2048 SHALL be '2', etc. When the job completes, the values of the  
1944 | **jmJobKOctetsRequested** object and the **jobKOctetsTransferred** attribute SHALL be  
1945 | equal.

1946 |  
1947 | NOTE - The **jobKOctetsTransferred** can be used with the **jmJobKOctetsRequested**  
1948 | object in order to produce a relative indication of the progress of the job for agents that do  
1949 | not implement the **jmJobKOctetsProcessed** object.

1950 |  
1951 |  
1952 | ++++++  
1953 | + **Impression attributes**  
1954 | +  
1955 | + **For a print job, an impression is the marking of the**  
1956 | + **entire side of a sheet. Two-sided processing involves two**  
1957 | + **impressions per sheet. Two-up is the placement of two**  
1958 | + **logical pages on one side of a sheet and so is still a**  
1959 | + **single impression. See also jmJobImpressionsRequested and**  
1960 | + **jmJobImpressionsCompleted objects in the jmJobTable.**  
1961 | ++++++

1962 |  
1963 | **impressionsSpooled(110), Integer32(-2..2147483647)**  
1964 | INTEGER: The number of impressions spooled to the server or device for the job so far.

1965 |  
1966 | **impressionsSentToDevice(111), Integer32(-2..2147483647)**  
1967 | INTEGER: The number of impressions sent to the device for the job so far.

1968 |

1969 **impressionsInterpreted(112), Integer32(-2..2147483647)**  
1970 INTEGER: The number of impressions interpreted for the job so far.

1971  
1972 **impressionsCompletedCurrentCopy(113), Integer32(-2..2147483647)**  
1973 INTEGER: The number of impressions completed by the device for the current copy of  
1974 the current document so far. For printing, the impressions completed includes  
1975 interpreting, marking, and stacking the output. For other types of job services, the  
1976 number of impressions completed includes the number of impressions processed.

1977  
1978 This value SHALL be reset to 0 for each document in the job and for each document  
1979 copy.

1980  
1981 **fullColorImpressionsCompleted(114), Integer32(-2..2147483647)**  
1982 INTEGER: The number of full color impressions completed by the device for this job so  
1983 far. For printing, the impressions completed includes interpreting, marking, and stacking  
1984 the output. For other types of job services, the number of impressions completed includes  
1985 the number of impressions processed. Full color impressions are typically defined as those  
1986 requiring 3 or more colorants, but this MAY vary by implementation.

1987  
1988 **highlightColorImpressionsCompleted(115), Integer32(-2..**  
1989 **2147483647)**  
1990 INTEGER: The number of highlight color impressions completed by the device for this  
1991 job so far. For printing, the impressions completed includes interpreting, marking, and  
1992 stacking the output. For other types of job services, the number of impressions completed  
1993 includes the number of impressions processed. Highlight color impressions are typically  
1994 defined as those requiring black plus one other colorant, but this MAY vary by  
1995 implementation.

1996  
1997  
1998 ++++++  
1999 + **Page attributes**  
2000 +  
2001 + **A page is a logical page. Number up can impose more than**  
2002 + **one page on a single side of a sheet. Two-up is the**  
2003 + **placement of two logical pages on one side of a sheet so**  
2004 + **that each side counts as two pages.**  
2005 ++++++

2006  
2007 **pagesRequested(130), Integer32(-2..2147483647)**  
2008 INTEGER: The number of logical pages requested by the job to be processed.

2009  
2010 **pagesCompleted(131), Integer32(-2..2147483647)**  
2011 INTEGER: The number of logical pages completed for this job so far.

2012  
2013 For implementations where multiple copies are produced by the interpreter with only a  
2014 single pass over the data, the final value SHALL be equal to the value of the  
2015 pagesRequested object. For implementations where multiple copies are produced by the  
2016 interpreter by processing the data for each copy, the final value SHALL be a multiple of  
2017 the value of the pagesRequested object.

2018  
2019  
2020  
2021  
2022  
2023  
2024  
2025  
2026  
2027  
2028  
2029  
2030  
2031  
2032  
2033  
2034  
2035  
2036  
2037  
2038  
2039  
2040  
2041  
2042  
2043  
2044  
2045  
2046  
2047  
2048  
2049  
2050  
2051  
2052  
2053  
2054  
2055  
2056  
2057  
2058  
2059  
2060  
2061  
2062  
2063  
2064  
2065  
2066

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

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

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

++++  
+ **Sheet attributes**  
+  
+ **The sheet is a single piece of a medium, whether printing on one or both sides.**  
++++

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

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

**sheetsCompletedCurrentCopy(152), Integer32(-2..2147483647)**  
INTEGER: The number of medium sheets that have completed marking and stacking for the current copy of a document in the job so far whether those sheets have been processed on one side or on both.  
  
The value of this attribute SHALL be reset to 0 as each document in the job starts being processed and for each document copy as it starts being processed.

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

**mediumRequested(170), JmMediumTypeTC AND/OR JmJobStringTC OCTET**

2067 STRING(SIZE(0..63))  
 2068 INTEGER: MULTI-ROW: The type  
 2069 AND/OR  
 2070 OCTETS: the name of the medium that is required by the job.  
 2071  
 2072 **mediumConsumed(171),** Integer32(-2..2147483647)  
 2073 AND  
 2074 JmJobStringTCOCTET  
 2075 STRING(SIZE(0..63))  
 2076 INTEGER: The number of sheets  
 2077 AND  
 2078 OCTETS: MULTI-ROW: the name of the medium that hasve been consumed so far  
 2079 whether those sheets have been processed on one side or on both.  
 2080

2081 This attribute SHALL have both **Integer32** and **OCTET STRING** (represented as  
 2082 JmJobStringTC) values.  
 2083

2084 **colorantRequested(172),** Integer32(-2..2147483647)  
 2085 AND/OR  
 2086 JmJobStringTCOCTET  
 2087 STRING(SIZE(0..63))  
 2088 INTEGER: MULTI-ROW: The index (**prtMarkerColorantIndex**) in the Printer  
 2089 MIB[print-mib]  
 2090 AND/OR  
 2091 OCTETS: the name of the colorant requested.  
 2092

2093 **colorantConsumed(173),** Integer32(-2..2147483647)  
 2094 AND/OR  
 2095 JmJobStringTCOCTET  
 2096 STRING(SIZE(0..63))  
 2097 INTEGER: MULTI-ROW: The index (**prtMarkerColorantIndex**) in the Printer  
 2098 MIB[print-mib]  
 2099 AND/OR  
 2100 OCTETS: the name of the colorant consumed.  
 2101

2102  
 2103 ++++++  
 2104 + **Time attributes (set by server or device)**  
 2105 +  
 2106 + **This section of attributes are ones that are set by the**  
 2107 + **server or device that accepts jobs. Two forms of time are**  
 2108 + **provided. Each form is represented in a separate attribute.**  
 2109 + **See section 3.1.2 and section 3.1.3 for the**  
 2110 + **conformance requirements for time attribute for agents and**  
 2111 + **monitoring applications, respectively. The two forms are:**  
 2112 +  
 2113 + **'DateAndTime' is an 8 or 11 octet binary encoded year,**  
 2114 + **month, day, hour, minute, second, deci-second with**  
 2115 + **optional offset from UTC. See SNMPv2-TC [SMIv2-TC].**

2116 +  
 2117 + **NOTE: 'DateAndTime' is not printable characters; it is**  
 2118 + **binary.**  
 2119 +  
 2120 + **'JmTimeStampTC' is the time of day measured in the number of**  
 2121 + **seconds since the system was booted.**  
 2122 ++++++

2124 **jobSubmissionToServerTime(190),** **JmTimeStampTC**  
 2125 **AND/OR**  
 2126 **DateAndTime**  
 2127 INTEGER: Configuration 3 only: The time  
 2128 AND/OR  
 2129 OCTETS: the date and time that the job was submitted to the server (as distinguished  
 2130 from the device which uses jobSubmissionTime).  
 2131

2132 **jobSubmissionTime(191),** **JmTimeStampTC**  
 2133 **AND/OR**  
 2134 **DateAndTime**  
 2135 INTEGER: Configurations 1, 2, and 3: The time  
 2136 AND/OR  
 2137 OCTETS: the date and time that the job was submitted to the server or device to which  
 2138 the agent is providing access.  
 2139

2142 **jobStartedBeingHeldTime(192),** **JmTimeStampTC**  
 2143 **AND/OR**  
 2144 **DateAndTime**  
 2145 INTEGER: The time  
 2146 AND/OR  
 2147 OCTETS: the date and time that the job last entered the **pendingHeld** state. If the job  
 2148 has never entered the **pendingHeld** state, then the value SHALL be '0' or the attribute  
 2149 SHALL not be present in the table.  
 2150

2151 **jobStartedProcessingTime(193),** **JmTimeStampTC**  
 2152 **AND/OR**  
 2153 **DateAndTime**  
 2154 INTEGER: The time  
 2155 AND/OR  
 2156 OCTETS: the date and time that the job started processing.  
 2157

2158 **jobCompletedTime(194),** **JmTimeStampTC**  
 2159 **AND/OR**  
 2160 **DateAndTime**  
 2161 INTEGER: The time  
 2162 AND/OR  
 2163 OCTETS: the date and time that the job entered the **completed, canceled, or aborted**  
 2164 state.





```

2214     printQualityRequested(70),
2215     printQualityUsed(71),
2216     printerResolutionRequested(72),
2217     printerResolutionUsed(73),
2218     tonerEcomonyRequested(74),
2219     tonerEcomonyUsed(75),
2220     tonerDensityRequested(76),
2221     tonerDensityUsed(77),
2222
2223     jobCopiesRequested(90),
2224     jobCopiesCompleted(91),
2225     documentCopiesRequested(92),
2226     documentCopiesCompleted(93),
2227     jobKOctetsTransferred(94),
2228
2229     impressionsSpooled(110),
2230     impressionsSentToDevice(111),
2231     impressionsInterpreted(112),
2232     impressionsCompletedCurrentCopy(113),
2233     fullColorImpressionsCompleted(114),
2234     highlightColorImpressionsCompleted(115),
2235
2236     pagesRequested(130),
2237     pagesCompleted(131),
2238     pagesCompletedCurrentCopy(132),
2239
2240     sheetsRequested(150),
2241     sheetsCompleted(151),
2242     sheetsCompletedCurrentCopy(152),
2243
2244     mediumRequested(170),
2245     mediumConsumed(171),
2246     colorantRequested(172),
2247     colorantConsumed(173),
2248
2249     jobSubmissionToServerTime(190),
2250     jobSubmissionTime(191),
2251     jobStartedBeingHeldTime(192),
2252     jobStartedProcessingTime(193),
2253     jobCompletedTime(194),
2254     jobProcessingCPUTime(195)
2255 }

```

```

2256
2257
2258
2259
2260 JmJobServiceTypesTC ::= TEXTUAL-CONVENTION
2261     STATUS current

```

2262	DESCRIPTION
2263	"Specifies the type(s) of service to which the job has been submitted (print, fax, scan, etc.). The
2264	service type is represented as an enum that is bit encoded with each job service type so that
2265	more general and arbitrary services can be created, such as services with more than one
2266	destination type, or ones with only a source or only a destination. For example, a job service
2267	might <b>scan</b> , <b>faxOut</b> , and <b>print</b> a single job. In this case, three bits would be set in the
2268	<b>jobServiceTypes</b> attribute, corresponding to the hexadecimal values: <b>0x8</b> + <b>0x20</b> + <b>0x4</b> ,
2269	respectively, yielding: <b>0x2C</b> .
2270	
2271	Whether this attribute is set from a job attribute supplied by the job submission client or is set by
2272	the recipient job submission server or device depends on the job submission protocol. With
2273	either implementation, the agent SHALL return a non-zero value for this attribute indicating the
2274	type of the job.
2275	
2276	One of the purposes of this attribute is to permit a requester to filter out jobs that are not of
2277	interest. For example, a printer operator MAY only be interested in jobs that include printing.
2278	That is why the attribute is in the job identification category.
2279	
2280	The following service component types are defined (in hexadecimal) and are assigned a separate
2281	bit value for use with the <b>jobServiceTypes</b> attribute:
2282	
2283	<b>other 0x1</b>
2284	The job contains some instructions that are not one of the identified types.
2285	
2286	<b>unknown 0x2</b>
2287	The job contains some instructions whose type is unknown to the agent.
2288	
2289	<b>print 0x4</b>
2290	The job contains some instructions that specify printing
2291	
2292	<b>scan 0x8</b>
2293	The job contains some instructions that specify scanning
2294	
2295	<b>faxIn 0x10</b>
2296	The job contains some instructions that specify receive fax
2297	
2298	<b>faxOut 0x20</b>
2299	The job contains some instructions that specify sending fax
2300	
2301	<b>getFile 0x40</b>
2302	The job contains some instructions that specify accessing files or documents
2303	
2304	<b>putFile 0x80</b>
2305	The job contains some instructions that specify storing files or documents
2306	
2307	<b>mailList 0x100</b>
2308	The job contains some instructions that specify distribution of documents using an
2309	electronic mail system."
2310	REFERENCE

2311 "These bit definitions are the equivalent of a type 2 enum except that combinations of them  
 2312 MAY be used together. See section 3.6.1.2."  
 2313 SYNTAX INTEGER(0..2147483647) -- 31 bits, all but sign bit

2314

2315

2316

2317

2318 **JmJobStateReasons1TC ::= TEXTUAL-CONVENTION**

2319 STATUS current

2320 DESCRIPTION

2321 "The **JmJobStateReasonsNTC** ( $N=1..4$ ) textual-conventions are used with the  
 2322 **jmJobStateReasons1** object and **jobStateReasonsN** ( $N=2..4$ ), respectively, to provide  
 2323 additional information regarding the current **jmJobState** object value. These values MAY be  
 2324 used with any job state or states for which the reason makes sense.

2325  
 2326 NOTE - While values cannot be added to the **jmJobState** object without impacting deployed  
 2327 clients that take actions upon receiving **jmJobState** values, it is the intent that additional  
 2328 **JmJobStateReasonsNTC** enums can be defined and registered without impacting such  
 2329 deployed clients. In other words, the **jmJobStateReasons1** object and **jobStateReasonsN**  
 2330 attributes are intended to be extensible.

2331  
 2332 NOTE - The Job Monitoring MIB contains a superset of the IPP values[ipp-model] for the IPP  
 2333 'job-state-reasons' attribute, since the Job Monitoring MIB is intended to cover other job  
 2334 submission protocols as well. Also some of the names of the reasons have been changed from  
 2335 'printer' to 'device', since the Job Monitoring MIB is intended to cover additional types of  
 2336 devices, including input devices, such as scanners.

2337  
 2338 The following standard values are defined (in hexadecimal) as *powers of two*, since multiple  
 2339 values MAY be used at the same time. For ease of understanding, the  
 2340 **JmJobStateReasons1TC** reasons are presented in the order in which the reasons are likely to  
 2341 occur (if implemented), starting with the '**jobIncoming**' value and ending with the  
 2342 '**jobCompletedWithErrors**' valuereasons.

2343  
 2344 **other** **0x1**  
 2345 The job state reason is not one of the standardized or registered reasons.

2346  
 2347 **unknown** **0x2**  
 2348 The job state reason is not known to the agent or is indeterminent.

2349  
 2350 **jobIncoming** **0x4**  
 2351 The job has been accepted by the server or device, but the server or device is expecting  
 2352 (1) additional operations from the client to finish creating the job and/or (2) is  
 2353 accessing/accepting document data.

2354  
 2355 **jobOutgoing** **0x8**  
 2356 Configuration 2 only: The server is transmitting the job to the device.

2357		
2358	<b>jobHoldSpecified</b>	<b>0x10</b>
2359	The value of the job's <b>jobHold(52)</b> attribute is TRUE. The job SHALL NOT be a	
2360	candidate for processing until this reason is removed and there are no other reasons to	
2361	hold the job.	
2362		
2363	<b>jobHoldUntilSpecified</b>	<b>0x20</b>
2364	The value of the job's <b>jobHoldUntil(53)</b> attribute specifies a time period that is still in the	
2365	future. The job SHALL NOT be a candidate for processing until this reason is removed	
2366	and there are no other reasons to hold the job.	
2367		
2368	<b>jobProcessAfterSpecified</b>	<b>0x40</b>
2369	The value of the job's <b>jobProcessAfterDateAndTime(51)</b> attribute specifies a time that is	
2370	still in the future. The job SHALL NOT be a candidate for processing until this reason is	
2371	removed and there are no other reasons to hold the job.	
2372		
2373	<b>resourcesAreNotReady</b>	<b>0x80</b>
2374	At least one of the resources needed by the job, such as media, fonts, resource objects,	
2375	etc., is not ready on any of the physical devices for which the job is a candidate. This	
2376	condition MAY be detected when the job is accepted, or subsequently while the job is	
2377	<b>pending</b> or <b>processing</b> , depending on implementation.	
2378		
2379	<b>deviceStoppedPartly</b>	<b>0x100</b>
2380	One or more, but not all, of the devices to which the job is assigned are stopped. If all of	
2381	the devices are stopped (or the only device is stopped), the <b>deviceStopped</b> reason	
2382	SHALL be used.	
2383		
2384	<b>deviceStopped</b>	<b>0x200</b>
2385	The device(s) to which the job is assigned is (are all) stopped.	
2386		
2387	<b>jobPrinting</b>	<b>0x400</b>
2388	The output device is marking media. This attribute is useful for servers and output devices	
2389	which spend a great deal of time processing when no marking is happening and then want	
2390	to show that marking is now happening or when the job is in the <b>canceled</b> or <b>aborted</b>	
2391	state, but the marking has not yet stopped so that impression or sheet counts are still	
2392	increasing for the job.	
2393		
2394	<b>jobCanceledByUser</b>	<b>0x800</b>
2395	The job was canceled by the user, i.e., by an unknown user or by a user whose name is the	
2396	same as the value of the job's <b>jmJobOwner</b> object.	
2397		
2398	<b>jobCanceledByOperator</b>	<b>0x1000</b>
2399	The job was canceled by the operator, i.e., by a user whose name is different than the	
2400	value of the job's <b>jmJobOwner</b> object.	
2401		
2402	<b>abortedBySystem</b>	<b>0x2000</b>
2403	The job was aborted by the system.	
2404		

2405 NOTE - When the system puts a job into the 'aborted' job state, this reason is not needed.  
 2406 This reason is needed only when the system aborts a job, but, instead of placing the job in  
 2407 the **aborted** job state, places the job in the **pendingHeld** state, so that a user or operator  
 2408 can manually try the job again.  
 2409

2410 **processingToStopPoint** **0x4000**

2411 The requester has issued an operation to cancel or interrupt the job or the server/device  
 2412 has aborted the job but the server/device is still performing some actions on the job until a  
 2413 specified stop point occurs or job termination/cleanup is completed.  
 2414

2415 This reason is recommended to be used in conjunction with the **canceled** or **aborted** job  
 2416 state to indicate that the server/device is still performing some actions on the job after the  
 2417 job leaves the **processing** state, so that some of the jobs resources consumed counters  
 2418 may still be incrementing while the job is in the **canceled** or **aborted** job states.  
 2419

2420 **jobCompletedSuccessfully** **0x84000**

2421 The job completed successfully.  
 2422

2423 **jobCompletedWithWarnings** **0x108000**

2424 The job completed with warnings.  
 2425

2426 **jobCompletedWithErrors** **0x210000**

2427 The job completed with errors (and possibly warnings too).  
 2428  
 2429

2430 The following additional job state reasons have been added to represent job states that are in  
 2431 ISO DPA[iso-dpa] and other job submission protocols:  
 2432

2433 **jobPaused** **0x420000**

2434 The job has been indefinitely suspended by a client issuing an operation to suspend the job  
 2435 so that other jobs may proceed using the same devices. The client MAY issue an  
 2436 operation to resume the paused job at any time, in which case the agent SHALL remove  
 2437 the **jobPaused** values from the job's **jmJobStateReasons1** object and the job is eventually  
 2438 resumed at or near the point where the job was paused.  
 2439

2440 **jobInterrupted** **0x840000**

2441 The job has been interrupted while processing by a client issuing an operation that  
 2442 specifies another job to be run instead of the current job. The server or device will  
 2443 automatically resume the interrupted job when the interrupting job completes.  
 2444

2445 **jobRetained** **0x1080000**

2446 The job is being retained by the server or device with all of the job's document data (and  
 2447 submitted resources, such as fonts, logos, and forms, if any). Thus a client could issue an  
 2448 operation to the server or device to either (1) re-do the job (or a copy of the job) on the  
 2449 same server or device or (2) resubmit the job to another server or device. When a client  
 2450 could no longer re-do/resubmit the job, such as after the document data has been  
 2451 discarded, the agent SHALL remove the **jobRetained** value from the  
 2452 **jmJobStateReasons1** object."  
 2453

REFERENCE

2454 "These bit definitions are the equivalent of a type 2 enum except that combinations of bits may  
 2455 be used together. See section 3.6.1.2. The remaining bits are reserved for future  
 2456 standardization and/or registration."  
 2457

2458 SYNTAX INTEGER(0..2147483647) -- 31 bits, all but sign bit  
 2459  
 2460  
 2461  
 2462  
 2463

2464 **JmJobStateReasons2TC ::= TEXTUAL-CONVENTION**

2465 STATUS current

2466 DESCRIPTION

2467 "This textual-convention is used with the **jobStateReasons2** attribute to provides additional  
 2468 information regarding the **jmJobState** object. See the description under  
 2469 **JmJobStateReasons1TC** for additional information that applies to all reasons.  
 2470

2471 The following standard values are defined (in hexadecimal) as *powers of two*, since multiple  
 2472 values may be used at the same time:  
 2473

2474 **cascaded** **0x1**

2475 An outbound gateway has transmitted all of the job's job and document attributes and data  
 2476 to another spooling system.  
 2477

2478 **deletedByAdministrator** **0x2**

2479 The administrator has deleted the job.  
 2480

2481 **discardTimeArrived** **0x4**

2482 The job has been deleted due to the fact that the time specified by the job's job-discard-  
 2483 time attribute has arrived.  
 2484

2485 **postProcessingFailed** **0x8**

2486 The post-processing agent failed while trying to log accounting attributes for the job;  
 2487 therefore the job has been placed into the completed state with the **jobRetained**  
 2488 **jmJobStateReasons1** object value for a system-defined period of time, so the  
 2489 administrator can examine it, resubmit it, etc.  
 2490

2491 **submissionInterrupted** **0x10**

2492 Indicates that the job was not completely submitted for some unforeseen reason, such as:  
 2493 (1) the server has crashed before the job was closed by the client, (2) the server or the  
 2494 document transfer method has crashed in some non-recoverable way before the document  
 2495 data was entirely transferred to the server, (3) the client crashed or failed to close the job  
 2496 before the time-out period.  
 2497

2498 **maxJobFaultCountExceeded** **0x20**

2499 The job has faulted several times and has exceeded the administratively defined fault count  
 2500 limit.  
 2501

2502	<b>devicesNeedAttentionTimeOut</b>	<b>0x40</b>
2503	One or more document transforms that the job is using needs human intervention in order	
2504	for the job to make progress, but the human intervention did not occur within the site-	
2505	settable time-out value.	
2506		
2507	<b>needsKeyOperatorTimeOut</b>	<b>0x80</b>
2508	One or more devices or document transforms that the job is using need a specially trained	
2509	operator (who may need a key to unlock the device and gain access) in order for the job to	
2510	make progress, but the key operator intervention did not occur within the site-settable	
2511	time-out value.	
2512		
2513	<b>jobStartWaitTimeOut</b>	<b>0x100</b>
2514	The server/device has stopped the job at the beginning of processing to await human	
2515	action, such as installing a special cartridge or special non-standard media, but the job was	
2516	not resumed within the site-settable time-out value and the server/device has transitioned	
2517	the job to the <b>pendingHeld</b> state.	
2518		
2519	<b>jobEndWaitTimeOut</b>	<b>0x200</b>
2520	The server/device has stopped the job at the end of processing to await human action,	
2521	such as removing a special cartridge or restoring standard media, but the job was not	
2522	resumed within the site-settable time-out value and the server/device has transitioned the	
2523	job to the completed state.	
2524		
2525	<b>jobPasswordWaitTimeOut</b>	<b>0x400</b>
2526	The server/device has stopped the job at the beginning of processing to await input of the	
2527	job's password, but the password was not received within the site-settable time-out value.	
2528		
2529	<b>deviceTimedOut</b>	<b>0x800</b>
2530	A device that the job was using has not responded in a period specified by the device's	
2531	site-settable attribute.	
2532		
2533	<b>connectingToDeviceTimeOut</b>	<b>0x1000</b>
2534	The server is attempting to connect to one or more devices which may be dial-up, polled,	
2535	or queued, and so may be busy with traffic from other systems, but server was unable to	
2536	connect to the device within the site-settable time-out value.	
2537		
2538	<b>transferring</b>	<b>0x2000</b>
2539	The job is being transferred to a down stream server or device.	
2540		
2541	<b>queuedInDevice</b>	<b>0x4000</b>
2542	The job has been queued in a down stream server or device.	
2543		
2544	<b>jobCleanup</b>	<b>0x8000</b>
2545	The server/device is performing cleanup activity as part of ending normal processing.	
2546		
2547	<del><b>processingToStopPoint</b></del>	<del><b>0x10000</b></del>
2548	<del>The requester has issued an operation to interrupt the job and the server/device is</del>	
2549	<del>processing up until the specified stop point occurs.</del>	
2550		

2551	<b>jobPasswordWait</b>	<b>0x20000</b>
2552	The server/device has selected the job to be next to process, but instead of assigning	
2553	resources and starting the job processing, the server/device has transitioned the job to the	
2554	<b>pendingHeld</b> state to await entry of a password (and dispatched another job, if there is	
2555	one).	
2556		
2557	<b>validating</b>	<b>0x40000</b>
2558	The server/device is validating the job <i>after</i> accepting the job.	
2559		
2560	<b>queueHeld</b>	<b>0x80000</b>
2561	The operator has held the entire job set or queue.	
2562		
2563	<b>jobProofWait</b>	<b>0x100000</b>
2564	The job has produced a single proof copy and is in the <b>pendingHeld</b> state waiting for the	
2565	requester to issue an operation to release the job to print normally, obeying any job and	
2566	document copy attributes that were originally submitted.	
2567		
2568	<b>heldForDiagnostics</b>	<b>0x200000</b>
2569	The system is running intrusive diagnostics, so that all jobs are being held.	
2570		
2571	<b>serviceOffLine</b>	<b>0x400000</b>
2572	The service/document transform is off-line and accepting no jobs. All pending jobs are put	
2573	into the pendingHeld state. This could be true if its input is impaired or broken.	
2574		
2575	<b>noSpaceOnServer</b>	<b>0x800000</b>
2576	There is no room on the server to store all of the job.	
2577		
2578	<b>pinRequired</b>	<b>0x1000000</b>
2579	The System Administrator settable device policy is (1) to require PINs, and (2) to hold	
2580	jobs that do not have a pin supplied as an input parameter when the job was created.	
2581		
2582	<b>exceededAccountLimit</b>	<b>0x2000000</b>
2583	The account for which this job is drawn has exceeded its limit. This condition <b>SHOULD</b>	
2584	be detected before the job is scheduled so that the user does not wait until his/her job is	
2585	scheduled only to find that the account is overdrawn. This condition <b>MAY</b> also occur	
2586	while the job is processing either as processing begins or part way through processing.	
2587		
2588	<b>heldForRetry</b>	<b>0x4000000</b>
2589	The job encountered some errors that the server/device could not recover from with its	
2590	normal retry procedures, but the error might not be encountered if the job is processed	
2591	again in the future. Example cases are phone number busy or remote file system in-	
2592	accessible. For such a situation, the server/device <b>SHALL</b> transition the job from the	
2593	<b>processing</b> to the <b>pendingHeld</b> , rather than to the <b>aborted</b> state.	
2594		
2595	The following values are from the X/Open PSIS draft standard:	
2596		
2597	<b>canceledByShutdown</b>	<b>0x8000000</b>
2598	The job was canceled because the server or device was shutdown before completing the	
2599	job.	





2649 **JmJobStateReasons4TC** ::= TEXTUAL-CONVENTION  
2650 STATUS current  
2651 DESCRIPTION  
2652 "This textual-convention is used in the **jobStateReasons4** attribute to provides additional  
2653 information regarding the **jmJobState** object. See the description under  
2654 **JmJobStateReasons1TC** for additional information that applies to all reasons.  
2655  
2656 The following standard values are defined (in hexadecimal) as *powers of two*, since multiple  
2657 values may be used at the same time:  
2658  
2659 none yet defined. These bits are reserved for future standardization and/or registration."  
2660 REFERENCE  
2661 "These bit definitions are the equivalent of a type 2 enum except that combinations of them may  
2662 be used together. See section 3.6.1.2. See the description under **JmJobStateReasons1TC** and  
2663 the **jobStateReasons4** attribute."  
2664  
2665 SYNTAX INTEGER(0..2147483647) -- 31 bits, all but sign bit

```

2666
2667 jobmonMIBObjects OBJECT IDENTIFIER ::= { jobmonMIB 1 }
2668
2669 -- The General Group (MANDATORY)
2670
2671 -- The jmGeneralGroup consists entirely of the jmGeneralTable.
2672
2673 jmGeneral OBJECT IDENTIFIER ::= { jobmonMIBObjects 1 }
2674
2675 jmGeneralTable OBJECT-TYPE
2676     SYNTAX      SEQUENCE OF JmGeneralEntry
2677     MAX-ACCESS  not-accessible
2678     STATUS      current
2679     DESCRIPTION
2680         "The jmGeneralTable consists of information of a general nature that are per-job-set, but are
2681         not per-job. See Section 2 entitled 'Terminology and Job Model' for the definition of a job set."
2682     REFERENCE
2683         "The MANDATORY-GROUP macro specifies that this group is MANDATORY."
2684     ::= { jmGeneral 1 }
2685
2686 jmGeneralEntry OBJECT-TYPE
2687     SYNTAX      JmGeneralEntry
2688     MAX-ACCESS  not-accessible
2689     STATUS      current
2690     DESCRIPTION
2691         "Information about a job set (queue).
2692
2693         An entry SHALL exist in this table for each job set."
2694     INDEX { jmGeneralJobSetIndex }
2695     ::= { jmGeneralTable 1 }
2696
2697 JmGeneralEntry ::= SEQUENCE {
2698     jmGeneralJobSetIndex           Integer32(1..32767),
2699     jmGeneralNumberOfActiveJobs   Integer32(0..2147483647),
2700     jmGeneralOldestActiveJobIndex Integer32(0..2147483647),
2701     jmGeneralNewestActiveJobIndex Integer32(0..2147483647),
2702     jmGeneralJobPersistence       Integer32(15..2147483647),
2703     jmGeneralAttributePersistence Integer32(15..2147483647),
2704     jmGeneralJobSetName           JmUTF8StringTCOCTET
2705     STRING(SIZE(0..63))
2706 }
2707
2708 jmGeneralJobSetIndex OBJECT-TYPE
2709     SYNTAX      Integer32(1..32767)
2710     MAX-ACCESS  not-accessible
2711     STATUS      current
2712     DESCRIPTION
2713         "A unique value for each job set in this MIB. The jmJobTable and jmAttributeTable tables
2714         have this same index as their primary index.

```

2715  
 2716 The value(s) of the **jmGeneralJobSetIndex** SHALL be persistent across power cycles, so that  
 2717 clients that have retained **jmGeneralJobSetIndex** values will access the same job sets upon  
 2718 subsequent power-up.  
 2719

2720 An implementation that has only one job set, such as a printer with a single queue, SHALL hard  
 2721 code this object with the value **1**."

2722 REFERENCE  
 2723 "See Section 2 entitled 'Terminology and Job Model' for the definition of a job set.  
 2724 Corresponds to the first index in **jmJobTable** and **jmAttributeTable**."  
 2725 ::= { jmGeneralEntry 1 }  
 2726

2727 **jmGeneralNumberOfActiveJobs** OBJECT-TYPE  
 2728 SYNTAX Integer32(0..2147483647)  
 2729 MAX-ACCESS read-only  
 2730 STATUS current  
 2731 DESCRIPTION  
 2732 "The current number of 'active' jobs in the **jmJobIDTable**, **jmJobTable**, and  
 2733 **jmAttributeTable**, i.e., the total number of jobs that are in the **pending**, **processing**, or  
 2734 **processingStopped** states. See the **JmJobStateTC** textual-convention for the exact  
 2735 specification of the semantics of the job states."  
 2736 ::= { jmGeneralEntry 2 }  
 2737

2738 **jmGeneralOldestActiveJobIndex** OBJECT-TYPE  
 2739 SYNTAX Integer32 (0..2147483647)  
 2740 MAX-ACCESS read-only  
 2741 STATUS current  
 2742 DESCRIPTION  
 2743 "The **jmJobIndex** of the oldest job that is still in one of the 'active' states (**pending**, **processing**,  
 2744 or **processingStopped**). In other words, the index of the 'active' job that has been in the job  
 2745 tables the longest.  
 2746  
 2747 If there are no active jobs, the agent SHALL set the value of this object to **0**."  
 2748 REFERENCE  
 2749 "See Section 3.2 entitled 'The Job Tables and the Oldest Active and Newest Active Indexes' for  
 2750 a description of the usage of this object."  
 2751 ::= { jmGeneralEntry 3 }  
 2752

2753 **jmGeneralNewestActiveJobIndex** OBJECT-TYPE  
 2754 SYNTAX Integer32 (0..2147483647)  
 2755 MAX-ACCESS read-only  
 2756 STATUS current  
 2757 DESCRIPTION  
 2758 "The **jmJobIndex** of the newest job that is in one of the 'active' states (**pending**, **processing**, or  
 2759 **processingStopped**). In other words, the index of the 'active' job that has been most recently  
 2760 added to the **job tables**.  
 2761  
 2762 When all jobs become 'inactive', i.e., enter the **pendingHeld**, **completed**, **canceled**, or **aborted**  
 2763 states, the agent SHALL set the value of this object to **0**."

2764 REFERENCE  
 2765 "See Section 3.2 entitled 'The Job Tables and the Oldest Active and Newest Active Indexes' for  
 2766 a description of the usage of this object."  
 2767 ::= { jmGeneralEntry 4 }  
 2768

**jmGeneralJobPersistence** OBJECT-TYPE  
 2769 SYNTAX **Integer32(15..2147483647)**  
 2770 UNITS "seconds"  
 2771 MAX-ACCESS read-only  
 2772 STATUS current  
 2773 DESCRIPTION  
 2774 "The minimum time in seconds for this instance of the Job Set that an entry SHALL remain in  
 2775 the **jmJobIDTable** and **jmJobTable** after **processing** has *completed*, i.e., the minimum time in  
 2776 seconds starting when the job enters the **completed, canceled, or aborted** state.  
 2777  
 2778 Depending on implementation, the value of this object MAY be either: (1) set by the system  
 2779 administrator by means outside this specification or (2) fixed by the implementation.  
 2780  
 2781 This value SHALL be equal to or greater than the value of **jmGeneralAttributePersistence**.  
 2782 This value SHOULD be at least 60 which gives a monitoring application one minute in which to  
 2783 poll for job data."  
 2784 DEFVAL { 60 } -- one minute  
 2785 ::= { jmGeneralEntry 5 }  
 2786  
 2787

**jmGeneralAttributePersistence** OBJECT-TYPE  
 2788 SYNTAX **Integer32(15..2147483647)**  
 2789 UNITS "seconds"  
 2790 MAX-ACCESS read-only  
 2791 STATUS current  
 2792 DESCRIPTION  
 2793 "The minimum time in seconds for this instance of the Job Set that an entry SHALL remain in  
 2794 the **jmAttributeTable** after **processing** has *completed*, i.e., the time in seconds starting when  
 2795 the job enters the **completed, canceled, or aborted** state.  
 2796  
 2797 Depending on implementation, the value of this object MAY be either (1) set by the system  
 2798 administrator by means outside this specification or MAY be (2) fixed by the implementation.  
 2799  
 2800 This value SHOULD be at least 60 which gives a monitoring application one minute in which to  
 2801 poll for job data."  
 2802 DEFVAL { 60 } -- one minute  
 2803 ::= { jmGeneralEntry 6 }  
 2804  
 2805

**jmGeneralJobSetName** OBJECT-TYPE  
 2806 SYNTAX **JmUTF8StringTCOCTET STRING(SIZE(0..63))**  
 2807 MAX-ACCESS read-only  
 2808 STATUS current  
 2809 DESCRIPTION  
 2810 "The human readable name of this job set assigned by the system administrator (by means  
 2811 outside of this MIB). Typically, this name SHOULD be the name of the job queue. If a server  
 2812

2813 or device has only a single job set, this object can be the administratively assigned name of the  
2814 server or device itself. This name does not need to be unique, though each job set in a single  
2815 Job Monitoring MIB SHOULD have distinct names.  
2816

2817 NOTE - The purpose of this object is to help the user of the job monitoring application  
2818 distinguish between several job sets in implementations that support more than one job set."  
2819 REFERENCE  
2820 "See the OBJECT compliance macro for the minimum maximum length required for  
2821 conformance."  
2822 ::= { jmGeneralEntry 7 }  
2823  
2824  
2825  
2826  
2827  
2828 -- The Job ID Group (MANDATORY)  
2829  
2830 -- The **jmJobIDGroup** consists entirely of the **jmJobIDTable**.  
2831  
2832 jmJobID OBJECT IDENTIFIER ::= { jobmonMIBObjects 2 }  
2833  
2834 jmJobIDTable OBJECT-TYPE  
2835 SYNTAX SEQUENCE OF JmJobIDEntry  
2836 MAX-ACCESS not-accessible  
2837 STATUS current  
2838 DESCRIPTION  
2839 "The **jmJobIDTable** provides a correspondence map (1) between the job submission ID that a  
2840 client uses to refer to a job and (2) the **jmGeneralJobSetIndex** and **jmJobIndex** that the Job  
2841 Monitoring MIB agent assigned to the job and that are used to access the job in all of the other  
2842 tables in the MIB. If a monitoring application already knows the **jmGeneralJobSetIndex** and  
2843 the **jmJobIndex** of the job it is querying, that application NEED NOT use the **jmJobIDTable**."  
2844 REFERENCE  
2845 "The MANDATORY-GROUP macro specifies that this group is MANDATORY."  
2846 ::= { jmJobID 1 }  
2847  
2848 jmJobIDEntry OBJECT-TYPE  
2849 SYNTAX JmJobIDEntry  
2850 MAX-ACCESS not-accessible  
2851 STATUS current  
2852 DESCRIPTION  
2853 "The map from (1) the **jmJobSubmissionID** to (2) the **jmGeneralJobSetIndex** and  
2854 **jmJobIndex**.  
2855  
2856 An entry SHALL exist in this table for each job currently known to the agent for all job sets and  
2857 job states. Each job SHALL appear in one and only one job set."  
2858 INDEX { **jmJobSubmissionID** }  
2859 ::= { jmJobIDTable 1 }  
2860  
2861 JmJobIDEntry ::= SEQUENCE {

```

2862     jmJobSubmissionID                                OCTET STRING(SIZE(48)),
2863     jmJobIDJobSetIndex                                Integer32(1..32767),
2864     jmJobIDJobIndex                                  Integer32(1..2147483647)
2865 }
2866
2867 jmJobSubmissionID OBJECT-TYPE
2868     SYNTAX      OCTET STRING(SIZE(48))
2869     MAX-ACCESS  not-accessible
2870     STATUS      current
2871     DESCRIPTION
2872         "A quasi-unique 48-octet fixed-length string ID which identifies the job within a particular
2873         client-server environment. There are multiple formats for the jmJobSubmissionID. Each
2874         format SHALL be uniquely identified. See the JmJobSubmissionIDTypeTC textual convention.
2875         Each format SHALL be registered using the procedures of a type 2 enum. See section 3.6.3
2876         entitled: 'IANA Registration of Job Submission Id Formats'.
2877
2878         If the requester (client or server) does not supply a job submission ID in the job submission
2879         protocol, then the recipient (server or device) SHALL assign a job submission ID using any of
2880         the standard formats that have been reserved to agents and adding the final 8 octets to
2881         distinguish the ID from others submitted from the same requester.
2882
2883         The monitoring application, whether in the client or running separately, MAY use the job
2884         submission ID to help identify which jmJobIndex was assigned by the agent, i.e., in which row
2885         the job information is in the other tables.
2886
2887         NOTE - fixed-length is used so that a management application can use a shortened GetNext
2888         varbind (in SNMPv1 and SNMPv2) in order to get the next submission ID, disregarding the
2889         remainder of the ID in order to access jobs independent of the trailing identifier part, e.g., to get
2890         all jobs submitted by a particular jmJobOwner or submitted from a particular MAC address."
2891     REFERENCE
2892         "See the JmJobSubmissionIDTypeTC textual convention."
2893         "See APPENDIX B - Support of the Job Submission ID in Job Submission Protocols."
2894     ::= { jmJobIDEntry 1 }
2895
2896 jmJobIDJobSetIndex OBJECT-TYPE
2897     SYNTAX      Integer32(1..32767)
2898     MAX-ACCESS  read-only
2899     STATUS      current
2900     DESCRIPTION
2901         "This object contains the value of the jmGeneralJobSetIndex for the job with the
2902         jmJobSubmissionID value, i.e., the job set index of the job set in which the job was placed
2903         when that server or device accepted the job. This 16-bit value in combination with the
2904         jmJobIDJobIndex value permits the management application to access the other tables to
2905         obtain the job-specific objects for this job."
2906     REFERENCE
2907         "See jmGeneralJobSetIndex in the jmGeneralTable."
2908     ::= { jmJobIDEntry 2 }
2909
2910 jmJobIDJobIndex OBJECT-TYPE

```

```

2911 SYNTAX Integer32(1..2147483647)
2912 MAX-ACCESS read-only
2913 STATUS current
2914 DESCRIPTION
2915     "This object contains the value of the jmJobIndex for the job with the jmJobSubmissionID
2916     value, i.e., the job index for the job when the server or device accepted the job. This value, in
2917     combination with the jmJobIDJobSetIndex value, permits the management application to
2918     access the other tables to obtain the job-specific objects for this job."
2919 REFERENCE
2920     "See jmJobIndex in the jmJobTable."
2921 ::= { jmJobIDEntry 3 }
2922
2923
2924
2925
2926 -- The Job Group (MANDATORY)
2927
2928 -- The jmJobGroup consists entirely of the jmJobTable.
2929
2930 jmJob OBJECT IDENTIFIER ::= { jobmonMIBObjects 3 }
2931
2932 jmJobTable OBJECT-TYPE
2933     SYNTAX SEQUENCE OF JmJobEntry
2934     MAX-ACCESS not-accessible
2935     STATUS current
2936     DESCRIPTION
2937         "The jmJobTable consists of basic job state and status information for each job in a job set that
2938         (1) monitoring applications need to be able to access in a single SNMP Get operation, (2) that
2939         have a single value per job, and (3) that SHALL always be implemented."
2940     REFERENCE
2941         "The MANDATORY-GROUP macro specifies that this group is MANDATORY."
2942     ::= { jmJob 1 }
2943
2944 jmJobEntry OBJECT-TYPE
2945     SYNTAX JmJobEntry
2946     MAX-ACCESS not-accessible
2947     STATUS current
2948     DESCRIPTION
2949         "Basic per-job state and status information.
2950
2951         An entry SHALL exist in this table for each job, no matter what the state of the job is. Each job
2952         SHALL appear in one and only one job set."
2953     REFERENCE
2954         "See Section 3.2 entitled 'The Job Tables'."
2955     INDEX { jmGeneralJobSetIndex, jmJobIndex }
2956     ::= { jmJobTable 1 }
2957
2958 JmJobEntry ::= SEQUENCE {
2959     jmJobIndex Integer32(1..2147483647),

```



```

2960     jmJobState                JmJobStateTC,
2961     jmJobStateReasons1        JmJobStateReasons1TC,
2962     jmNumberOfInterveningJobs Integer32(-2..2147483647),
2963     jmJobKOctetsRequested     Integer32(-2..2147483647),
2964     jmJobKOctetsProcessed     Integer32(-2..2147483647),
2965     jmJobImpressionsRequested Integer32(-2..2147483647),
2966     jmJobImpressionsCompleted Integer32(-2..2147483647),
2967     jmJobOwner                JmJobStringTCOCTET
2968     STRING(SIZE(0..63))
2969 }
2970
2971 jmJobIndex OBJECT-TYPE
2972     SYNTAX      Integer32(1..2147483647)
2973     MAX-ACCESS not-accessible
2974     STATUS      current
2975     DESCRIPTION
2976         "The sequential, monotonically increasing identifier index for the job generated by the server or
2977         device when that server or device accepted the job. This index value permits the management
2978         application to access the other tables to obtain the job-specific row entries."
2979     REFERENCE
2980         "See Section 3.2 entitled 'The Job Tables and the Oldest Active and Newest Active Indexes'.
2981         See Section 3.4 entitled 'Job Identification'.
2982         See also jmGeneralNewestActiveJobIndex for the largest value of jmJobIndex.
2983         See JmJobSubmissionTypeTC for a limit on the size of this index if the agent represents it as
2984         an 8-digit decimal number."
2985     ::= { jmJobEntry 1 }
2986
2987 jmJobState OBJECT-TYPE
2988     SYNTAX      JmJobStateTC
2989     MAX-ACCESS read-only
2990     STATUS      current
2991     DESCRIPTION
2992         "The current state of the job (pending, processing, completed, etc.). Agents SHALL
2993         implement only those states which are appropriate for the particular implementation. However,
2994         management applications SHALL be prepared to receive all the standard job states.
2995
2996         The final value for this object SHALL be one of: completed, canceled, or aborted. The
2997         minimum length of time that the agent SHALL maintain MIB data for a job in the completed,
2998         canceled, or aborted state before removing the job data from the jmJobIDTable and
2999         jmJobTable is specified by the value of the jmGeneralJobPersistence object."
3000     ::= { jmJobEntry 2 }
3001
3002 jmJobStateReasons1 OBJECT-TYPE
3003     SYNTAX      JmJobStateReasons1TC
3004     MAX-ACCESS read-only
3005     STATUS      current
3006     DESCRIPTION
3007         "Additional information about the job's current state, i.e., information that augments the value of
3008         the job's jmJobState object.

```

3009  
 3010 Implementation of any reason values is OPTIONAL, but an agent SHOULD return any reason  
 3011 information available. These values MAY be used with any job state or states for which the  
 3012 reason makes sense. Furthermore, when implemented as with any MIB data, the agent SHALL  
 3013 return these values when the reason applies and SHALL NOT return them when the reason no  
 3014 longer applies whether the value of the job's **jmJobState** object changed or not. When the  
 3015 agent cannot provide a reason for the current state of the job, the agent SHALL set the value of  
 3016 the **jmJobStateReasons1** object and **jobStateReasonsN** attributes to **0**."

3017 REFERENCE  
 3018 "The **jobStateReasonsN** ( $N=2..4$ ) attributes provide further additional information about the  
 3019 job's current state."  
 3020 ::= { jmJobEntry 3 }

3021  
 3022 **jmNumberOfInterveningJobs** OBJECT-TYPE  
 3023 SYNTAX **Integer32(-2..2147483647)**  
 3024 MAX-ACCESS read-only  
 3025 STATUS current  
 3026 DESCRIPTION  
 3027 "The number of jobs that are expected to complete being processed *before* this job has  
 3028 completed being processed according to the implementation's queuing algorithm if no other  
 3029 jobs were to be submitted. In other words, this value is the job's queue position. The agent  
 3030 SHALL return a value of **0** for this attribute when while the job is the next job to complete  
 3031 processing (or has completed processing)."  
 3032 ::= { jmJobEntry 4 }

3033  
 3034 **jmJobKOctetsRequested** OBJECT-TYPE  
 3035 SYNTAX **Integer32(-2..2147483647)**  
 3036 MAX-ACCESS read-only  
 3037 STATUS current  
 3038 DESCRIPTION  
 3039 "The total size in K (1024) octets of the document(s) being requested to be processed in the job.  
 3040 The agent SHALL round the actual number of octets up to the next highest K. Thus 0 octets  
 3041 SHALL be represented as '0', 1-1024 octets SHALL be represented as '1', 1025-2048 SHALL  
 3042 be represented as '2', etc.  
 3043  
 3044 In computing this value, the server/device SHALL *not* include the multiplicative factors  
 3045 contributed by (1) the number of document copies, and (2) the number of job copies,  
 3046 independent of whether the device can process multiple copies of the job or document without  
 3047 making multiple passes over the job or document data and independent of whether the output is  
 3048 collated or not. Thus the server/device computation is independent of the implementation."  
 3049 ::= { jmJobEntry 5 }

3050  
 3051 **jmJobKOctetsProcessed** OBJECT-TYPE  
 3052 SYNTAX **Integer32(-2..2147483647)**  
 3053 MAX-ACCESS read-only  
 3054 STATUS current  
 3055 DESCRIPTION  
 3056 "The current number of octets processed by the server or device measured in units of K (1024)  
 3057 octets. The agent SHALL round the actual number of octets processed up to the next higher K."

3058 Thus 0 octets SHALL be represented as '0', 1-1024 octets SHALL be represented as '1', 1025-  
 3059 2048 octets SHALL be '2', etc. For printing devices, this value is the number interpreted by the  
 3060 page description language interpreter rather than what has been marked on media.  
 3061

3062 For implementations where multiple copies are produced by the interpreter with only a single  
 3063 pass over the data, the final value SHALL be equal to the value of the  
 3064 **jmJobKOctetsRequested** object. For implementations where multiple copies are produced by  
 3065 the interpreter by processing the data for each copy, the final value SHALL be a multiple of the  
 3066 value of the **jmJobKOctetsRequested** object.  
 3067

3068 NOTE - See the **impressionsCompletedCurrentCopy** and **pagesCompletedCurrentCopy**  
 3069 attributes for attributes that are reset on each document copy.  
 3070

3071 NOTE - The **jmJobKOctetsProcessed** object can be used with the **jmJobKOctetsRequested**  
 3072 object to provide an indication of the relative progress of the job, provided that the  
 3073 multiplicative factor is taken into account for some implementations of multiple copies."  
 3074 ::= { jmJobEntry 6 }

3075  
 3076 **jmJobImpressionsRequested** OBJECT-TYPE

3077 SYNTAX **Integer32(-2..2147483647)**

3078 MAX-ACCESS read-only

3079 STATUS current

3080 DESCRIPTION

3081 "The total size in number of impressions of the document(s) being requested by this job to  
 3082 produce.

3083  
 3084 In computing this value, the server/device SHALL not include the multiplicative factors  
 3085 contributed by (1) the number of document copies, and (2) the number of job copies,  
 3086 independent of whether the device can process multiple copies of the job or document without  
 3087 making multiple passes over the job or document data and independent of whether the output is  
 3088 collated or not. Thus the server/device computation is independent of the implementation."

3089 ::= { jmJobEntry 7 }

3090  
 3091 **jmJobImpressionsCompleted** OBJECT-TYPE

3092 SYNTAX **Integer32(-2..2147483647)**

3093 MAX-ACCESS read-only

3094 STATUS current

3095 DESCRIPTION

3096 "The current number of impressions completed for this job so far. For printing devices, the  
 3097 impressions completed includes interpreting, marking, and stacking the output. For other types  
 3098 of job services, the number of impressions completed includes the number of impressions  
 3099 processed.  
 3100

3101 For implementations where multiple copies are produced by the interpreter with only a single  
 3102 pass over the data, the final value SHALL be equal to the value of the  
 3103 **jmJobImpressionsRequested** object. For implementations where multiple copies are produced  
 3104 by the interpreter by processing the data for each copy, the final value SHALL be a multiple of  
 3105 the value of the **jmJobImpressionsRequested** object.  
 3106

3107 NOTE - See the **impressionsCompletedCurrentCopy** and **pagesCompletedCurrentCopy**  
 3108 attributes for attributes that are reset on each document copy.

3109

3110 NOTE - The **jmJobImpressionsCompleted** object can be used with the  
 3111 **jmJobImpressionsRequested** object to provide an indication of the relative progress of the job,  
 3112 provided that the multiplicative factor is taken into account for some implementations of  
 3113 multiple copies."

3114 ::= { jmJobEntry 8 }

3115

3116 **jmJobOwner** OBJECT-TYPE  
 3117 SYNTAX **JmJobStringTCOCTET STRING(SIZE(0..63))**  
 3118 MAX-ACCESS read-only  
 3119 STATUS current  
 3120 DESCRIPTION  
 3121 "The coded character set name of the user that submitted the job. The method of assigning this  
 3122 user name will be system and/or site specific but the method MUST insure that the name is  
 3123 unique to the network that is visible to the client and target device.  
 3124  
 3125 This value SHOULD be the *authenticated* name of the user submitting the job."  
 3126 REFERENCE  
 3127 "See the OBJECT compliance macro for the minimum maximum length required for  
 3128 conformance."  
 3129 ::= { jmJobEntry 9 }

3130

3131

3132

3133

3134 -- The Attribute Group (MANDATORY)

3135

3136 -- The **jmAttributeGroup** consists entirely of the **jmAttributeTable**.

3137 --

3138 -- Implementation of the two objects in this group is MANDATORY.

3139 -- See Section 3.1 entitled 'Conformance Considerations'.

3140 -- An agent SHALL implement any attribute if (1) the server or device

3141 -- supports the functionality represented by the attribute and (2) the

3142 -- information is available to the agent.

3143

3144 jmAttribute OBJECT IDENTIFIER ::= { jobmonMIBObjects 4 }

3145

3146 **jmAttributeTable** OBJECT-TYPE  
 3147 SYNTAX SEQUENCE OF JmAttributeEntry  
 3148 MAX-ACCESS not-accessible  
 3149 STATUS current  
 3150 DESCRIPTION  
 3151 "The **jmAttributeTable** SHALL contain attributes of the job and document(s) for each job in a  
 3152 job set. Instead of allocating distinct objects for each attribute, each attribute is represented as a  
 3153 separate row in the **jmAttributeTable**."  
 3154 REFERENCE

3155 "The MANDATORY-GROUP macro specifies that this group is MANDATORY. An agent  
 3156 SHALL implement any attribute if (1) the server or device supports the functionality represented  
 3157 by the attribute and (2) the information is available to the agent. "  
 3158 ::= { **jmAttribute** 1 }

3159

3160 **jmAttributeEntry** OBJECT-TYPE  
 3161 SYNTAX **JmAttributeEntry**  
 3162 MAX-ACCESS not-accessible  
 3163 STATUS current  
 3164 DESCRIPTION  
 3165 "Attributes representing information about the job and document(s) or resources required and/or  
 3166 consumed.  
 3167  
 3168 Each entry in the **jmAttributeTable** is a per-job entry with an extra index for each type of  
 3169 attribute (**jmAttributeTypeIndex**) that a job can have and an additional index  
 3170 (**jmAttributeInstanceIndex**) for those attributes that can have multiple instances per job. The  
 3171 **jmAttributeTypeIndex** object SHALL contain an enum type that indicates the type of attribute  
 3172 (see the **JmAttributeTypeTC** textual-convention). The value of the attribute SHALL be  
 3173 represented in either the **jmAttributeValueAsInteger** or **jmAttributeValueAsOctets** objects,  
 3174 and/or both, as specified in the **JmAttributeTypeTC** textual-convention.  
 3175  
 3176 The agent SHALL create rows in the **jmAttributeTable** as the server or device is able to  
 3177 discover the attributes either from the job submission protocol itself or from the document PDL.  
 3178 As the documents are interpreted, the interpreter MAY discover additional attributes and so the  
 3179 agent adds additional rows to this table. As the attributes that represent resources are actually  
 3180 consumed, the usage counter contained in the **jmAttributeValueAsInteger** object is  
 3181 incremented according to the units indicated in the description of the **JmAttributeTypeTC**  
 3182 enum.  
 3183  
 3184 The agent SHALL maintain each row in the **jmJobTable** for at least the minimum time after a  
 3185 job completes as specified by the **jmGeneralAttributePersistence** object.  
 3186  
 3187 Zero or more entries SHALL exist in this table for each job in a job set."  
 3188 REFERENCE  
 3189 "See Section 3.3 entitled 'The Attribute Mechanism' for a description of the **jmAttributeTable**."  
 3190 INDEX { **jmGeneralJobSetIndex**, **jmJobIndex**, **jmAttributeTypeIndex**,  
 3191 **jmAttributeInstanceIndex** }  
 3192 ::= { **jmAttributeTable** 1 }

3193

3194 **JmAttributeEntry** ::= SEQUENCE {  
 3195 **jmAttributeTypeIndex** **JmAttributeTypeTC**,  
 3196 **jmAttributeInstanceIndex** **Integer32(1..32767)**,  
 3197 **jmAttributeValueAsInteger** **Integer32(-2..2147483647)**,  
 3198 **jmAttributeValueAsOctets** **OCTET STRING(SIZE(0..63))**  
 3199 }  
 3200

3201 **jmAttributeTypeIndex** OBJECT-TYPE  
 3202 SYNTAX **JmAttributeTypeTC**  
 3203 MAX-ACCESS not-accessible

3204 STATUS current  
 3205 DESCRIPTION  
 3206 "The type of attribute that this row entry represents.  
 3207  
 3208 The type MAY identify information about the job or document(s) or MAY identify a resource  
 3209 required to process the job before the job start processing and/or consumed by the job as the job  
 3210 is processed.  
 3211  
 3212 Examples of ~~job and document~~ attributes (i.e., apply to the job as a whole) that have only one  
 3213 instance per job include: **jobCopiesRequested(90)**, **documentCopiesRequested(92)**,  
 3214 **jobCopiesCompleted(91)**, **documentCopiesCompleted(93)**, while examples of job attributes  
 3215 that may have more than one instance per job include: **documentFormatIndex(37)**, and  
 3216 **documentFormat(38)**.  
 3217  
 3218 Examples of document attributes (one instance per document) include: **fileName(34)**, and  
 3219 **documentName(35)**.  
 3220  
 3221 Examples of required and consumed resource attributes include: **pagesRequested(130)**,  
 3222 **mediumRequested(170)**, **pagesCompleted(131)**, and **mediumConsumed(171)**, respectively."  
 3223 ::= { jmAttributeEntry 1 }  
 3224  
 3225 **jmAttributeInstanceIndex** OBJECT-TYPE  
 3226 SYNTAX **Integer32(1..32767)**  
 3227 MAX-ACCESS not-accessible  
 3228 STATUS current  
 3229 DESCRIPTION  
 3230 "A running 16-bit index of the attributes of the same type for each job. For those attributes with  
 3231 only a single instance per job, this index value SHALL be **1**. For those attributes that are a  
 3232 single value per document, the index value SHALL be the document number, starting with **1** for  
 3233 the first document in the job. Jobs with only a single document SHALL use the index value of  
 3234 **1**. For those attributes that can have multiple values per job or per document, such as  
 3235 **documentFormatIndex(37)** or **documentFormat(38)**, the index SHALL be a running index  
 3236 for the job as a whole, starting at **1**."  
 3237 ::= { jmAttributeEntry 2 }  
 3238  
 3239 **jmAttributeValueAsInteger** OBJECT-TYPE  
 3240 SYNTAX **Integer32(-2..2147483647)**  
 3241 MAX-ACCESS read-only  
 3242 STATUS current  
 3243 DESCRIPTION  
 3244 "The integer value of the attribute. The value of the attribute SHALL be represented as an  
 3245 integer if the enum description in the **JmAttributeTypeTC** textual-convention definition has the  
 3246 tag: 'INTEGER:'.  
 3247  
 3248 Depending on the enum definition, this object value MAY be an integer, a counter, an index, or  
 3249 an enum, depending on the **jmAttributeTypeIndex** value. The units of this value are specified  
 3250 in the enum description.  
 3251

3252 For those attributes that are accumulating job consumption as the job is processed as specified in  
 3253 the **JmAttributeTypeTC** textual-convention, SHALL contain the final value after the job  
 3254 completes processing, i.e., this value SHALL indicate the total usage of this resource made by  
 3255 the job.  
 3256

3257 A monitoring application is able to copy this value to a suitable longer term storage for later  
 3258 processing as part of an accounting system.  
 3259

3260 Since the agent MAY add attributes representing resources to this table while the job is waiting  
 3261 to be processed or being processed, which can be a long time before any of the resources are  
 3262 actually used, the agent SHALL set the value of the **jmAttributeValueAsInteger** object to **0**  
 3263 for resources that the job has not yet consumed.  
 3264

3265 Attributes for which the concept of an integer value is meaningless, such as **fileName(34)**,  
 3266 **jobNameInterpreter**, and **processingMessagePhysicalDevice**, do *not* have the 'INTEGER:'  
 3267 tag in the **JmAttributeTypeTC** definition and so an agent SHALL always return a value of '-1'  
 3268 to indicate 'other' for the value of the **jmAttributeValueAsInteger** object for these attributes.  
 3269

3270 For attributes which do have the 'INTEGER:' tag in the **JmAttributeTypeTC** definition, if the  
 3271 integer value is not (yet) known, the agent either (1) SHALL not materialize the row in the  
 3272 **jmAttributeTable** until the value is known or (2) SHALL return a '-2' to represent an  
 3273 'unknown' counting integer value, a '0' to represent an 'unknown' index value, and a '2' to  
 3274 represent an 'unknown(2)' enum value."  
 3275 ::= { jmAttributeEntry 3 }

3276  
 3277 **jmAttributeValueAsOctets** OBJECT-TYPE  
 3278 SYNTAX OCTET STRING(SIZE(0..63))  
 3279 MAX-ACCESS read-only  
 3280 STATUS current  
 3281 DESCRIPTION

3282 "The octet string value of the attribute. The value of the attribute SHALL be represented as an  
 3283 OCTET STRING if the enum description in the **JmAttributeTypeTC** textual-convention  
 3284 definition has the tag: 'OCTETS:'.  
 3285

3286 Depending on the enum definition, this object value MAY be a coded character set string (text),  
 3287 such as '**JmUTF8StringTC**', or a binary octet string, such as '**DateAndTime**'.  
 3288

3289 Attributes for which the concept of an octet string value is meaningless, such as  
 3290 **pagesCompleted**, do *not* have the tag 'OCTETS:' in the **JmAttributeTypeTC** definition and so  
 3291 the agent SHALL always return a zero length string for the value of the  
 3292 **jmAttributeValueAsOctets** object.  
 3293

3294 For attributes which do have the 'OCTETS:' tag in the **JmAttributeTypeTC** definition, if the  
 3295 OCTET STRING value is not (yet) known, the agent either SHALL not materialize the row in  
 3296 the **jmAttributeTable** until the value is known or SHALL return a zero-length string."  
 3297 ::= { jmAttributeEntry 4 }

3298

```

3299 -- Notifications and Trapping
3300 -- Reserved for the future
3301
3302 jobmonMIBNotifications OBJECT IDENTIFIER ::= { jobmonMIB 2 }
3303
3304
3305
3306 -- Conformance Information
3307
3308 jmMIBConformance OBJECT IDENTIFIER ::= { jobmonMIB 3 }
3309
3310 -- compliance statements
3311 jmMIBCompliance MODULE-COMPLIANCE
3312     STATUS current
3313     DESCRIPTION
3314         "The compliance statement for agents that implement the
3315         job monitoring MIB."
3316     MODULE -- this module
3317     MANDATORY-GROUPS {
3318         jmGeneralGroup, jmJobIDGroup, jmJobGroup, jmAttributeGroup }
3319
3320     OBJECT jmGeneralJobSetName
3321     SYNTAX JmUTF8StringTCOCTET STRING (SIZE(0..8))
3322     DESCRIPTION
3323         "Only 8 octets maximum string length NEED be supported by the agent."
3324
3325     OBJECT jmJobOwner
3326     SYNTAX JmJobStringTCOCTET STRING (SIZE(0..16))
3327     DESCRIPTION
3328         "Only 16 octets maximum string length NEED be supported by the agent."
3329
3330 -- There are no CONDITIONALLY MANDATORY or OPTIONAL groups.
3331
3332     ::= { jmMIBConformance 1 }
3333
3334 jmMIBGroups OBJECT IDENTIFIER ::= { jmMIBConformance 2 }
3335
3336 jmGeneralGroup OBJECT-GROUP
3337     OBJECTS {
3338         jmGeneralNumberOfActiveJobs, jmGeneralOldestActiveJobIndex,
3339         jmGeneralNewestActiveJobIndex, jmGeneralJobPersistence,
3340         jmGeneralAttributePersistence, jmGeneralJobSetName }
3341     STATUS current
3342     DESCRIPTION
3343         "The general group."
3344     ::= { jmMIBGroups 1 }
3345
3346 jmJobIDGroup OBJECT-GROUP
3347     OBJECTS {

```



```
3348         jmJobIDJobSetIndex, jmJobIDJobIndex }
3349     STATUS current
3350     DESCRIPTION
3351         "The job ID group."
3352     ::= { jmMIBGroups 2 }
3353
3354     jmJobGroup OBJECT-GROUP
3355         OBJECTS {
3356             jmJobState, jmJobStateReasons1, jmNumberOfInterveningJobs,
3357             jmJobKOctetsRequested, jmJobKOctetsProcessed, jmJobImpressionsRequested,
3358             jmJobImpressionsCompleted, jmJobOwner }
3359     STATUS current
3360     DESCRIPTION
3361         "The job group."
3362     ::= { jmMIBGroups 3 }
3363
3364     jmAttributeGroup OBJECT-GROUP
3365         OBJECTS {
3366             jmAttributeValueAsInteger, jmAttributeValueAsOctets }
3367     STATUS current
3368     DESCRIPTION
3369         "The attribute group."
3370     ::= { jmMIBGroups 4 }
3371
3372
3373     END
```

## 3374 5. Appendix A - Implementing the Job Life Cycle

3375 The job object has well-defined states and client operations that affect the transition between the  
3376 job states. Internal server and device actions also affect the transitions of the job between the job  
3377 states. These states and transitions are referred to as the job's *life cycle*.

3378 Not all implementations of job submission protocols have all of the states of the job model  
3379 specified here. The job model specified here is intended to be a superset of most implementations.  
3380 It is the purpose of the agent to map the particular implementation's job life cycle onto the one  
3381 specified here. The agent MAY omit any states not implemented. Only the **processing** and  
3382 **completed** states are required to be implemented by an agent. However, a conforming  
3383 management application SHALL be prepared to accept any of the states in the job life cycle  
3384 specified here, so that the management application can interoperate with any conforming agent.

3385 The job states are intended to be user visible. The agent SHALL make these states visible in the  
3386 MIB, but only for the subset of job states that the implementation has. Some implementations  
3387 MAY need to have sub-states of these user-visible states. The **jmJobStateReasons1** object and  
3388 the **jobStateReasonsN** ( $N=2..4$ ) attributes can be used to represent the sub-states of the jobs.

3389 Job states are intended to last a user-visible length of time in most implementations. However,  
3390 some jobs may pass through some states in zero time in some situations and/or in some  
3391 implementations.

3392 The job model does not specify how accounting and auditing is implemented, except to assume  
3393 that accounting and auditing logs are separate from the job life cycle and last longer than job  
3394 entries in the MIB. Jobs in the **completed**, **aborted**, or **canceled** states are not logs, since jobs in  
3395 these states are accessible via SNMP protocol operations and SHALL be removed from the Job  
3396 Monitoring MIB tables after a site-settable or implementation-defined period of time. An  
3397 accounting application MAY copy accounting information incrementally to an accounting log as a  
3398 job processes, or MAY be copied while the job is in the **canceled**, **aborted**, or **completed** states,  
3399 depending on implementation. The same is true for auditing logs.

3400 **The jmJobState object specifies the standard job states. The normal job state transitions**  
3401 **are shown in the state transition diagram presented in Table 1.**

## 3402 6. APPENDIX B - Support of the Job Submission ID in Job Submission 3403 Protocols

3404 This appendix lists the job submission protocols that support the concept of a job  
3405 submission ID and indicates the attribute used in that job submission protocol.

3406 **6.1 Hewlett-Packard's Printer Job Language (PJL)**

3407 Hewlett-Packard's Printer Job Language provides job-level printer control and printer  
3408 status information to applications. The PJL JOB command is used at the beginning of a  
3409 print job and can include options applying only to that job. A PJL JOB command option  
3410 has been defined to facilitate passing the **JobSubmissionID** with the print job, as required  
3411 by the Job Monitoring MIB. The option is of the form:

```
3412         SUBMISSIONID = "id string"  
3413  
3414
```

3415 Where the "id string" is a string and SHALL be enclosed in double quotes. The format is  
3416 as described for the **jmJobSubmissionID** object.

3417 The entire PJL JOB command with the optional parameter would be of the form:

```
3418         @PJL JOB SUBMISSIONID = "id string"  
3419  
3420
```

3421 See "Printer Job Language Technical Reference Manual", part number 5021-0328, from  
3422 Hewlett-Packard for complete information on the PJL JOB command and the Printer Job  
3423 Language.

3424 NOTE - Some PJL implementations wrap a banner page as a PJL job around a job  
3425 submitted by a client. In this case, there will be two job submission ids. The outer one  
3426 being the one with the banner page and the inner one being the original user's job. The  
3427 agent SHALL use the last received job submission ID for the jmJobSubmissionID index,  
3428 so that the original user's job submission ID will be used, not the banner page job ID.

3429 **6.2 ISO DPA**

3430 The ISO 10175 Document Printing Application (DPA) protocol specifies the "**job-client-**  
3431 **id**" attribute that allows the client to supply a text string ID for each job.

3432 **7. References**

3433 [char-set policy] Harald Avelstrand, "IETF Policy on Character Sets and Language", June  
3434 1997. Latest draft: <draft-avelstrand-charset-policy-00.txt>

3435 [GB2312] GB 2312-1980, "Chinese People's Republic of China (PRC) mixed one byte  
3436 and two byte coded character set"

3437 [hr-mib] P. Grillo, S. Waldbusser, "Host Resources MIB", RFC 1514, September 1993

3438 [iana] J. Reynolds, and J. Postel, "Assigned Numbers", STD 2, RFC 1700, ISI, October  
3439 1994.

- 3440 [IANA-charsets] Coded Character Sets registered by IANA and assigned an enum value  
3441 for use in the **CodedCharSet** textual convention imported from the Printer MIB. See  
3442 <ftp://ftp.isi.edu/in-notes/iana/assignments/character-sets>
- 3443 [iana-media-types] IANA Registration of MIME media types (MIME content  
3444 types/subtypes). See <ftp://ftp.isi.edu/in-notes/iana/assignments/>
- 3445 [ISO 646] ISO/IEC 646:1991, "Information technology -- ISO 7-bit coded character set  
3446 for information interchange", JTC1/SC2.
- 3447 [ISO 8859] ISO/IEC 8859-1:1987, "Information technology -- 8-bit single byte coded  
3448 graphic character sets - Part 1: Latin alphabet No. 1, JTC1/SC2."
- 3449 [ISO 2022] ISO/IEC 2022:1994 - "Information technology -- Character code structure  
3450 and extension techniques", JTC1/SC2.
- 3451 [ISO-10646] ISO/IEC 10646-1:1993, "Information technology -- Universal Multiple-  
3452 Octet Coded Character Set (UCS) - Part 1: Architecture and Basic Multilingual Plane,  
3453 JTC1/SC2.
- 3454 [iso-dpa] ISO/IEC 10175 Document Printing Application (DPA). See  
3455 <ftp://ftp.pwg.org/pub/pwg/dpa/>
- 3456 [ipp-model] Internet Printing Protocol (IPP), work in progress on the IETF standards  
3457 track. See **draft-ietf-ipp-model-01.txt**. See also <http://www.pwg.org/ipp/index.html>
- 3458 [JIS X0208] JIS X0208-1990, "Japanese two byte coded character set."
- 3459 [mib-II] MIB-II, RFC 1213.
- 3460 [print-mib] The Printer MIB - RFC 1759, proposed IETF standard. Also an Internet-  
3461 Draft on the standards track as a draft standard: **draft-ietf-printmib-mib-info-02.txt**
- 3462 [req-words] S. Bradner, "Keywords for use in RFCs to Indicate Requirement Levels",  
3463 RFC 2119, March 1997.
- 3464 [rfc 2130] C. Weider, C. Preston, K. Simonsen, H. Alvestrand, R. Atkinson, M. Crispin,  
3465 and P. Svanberg, "The Report of the IAB Character Set Workshop held 29 Feb-1 March,  
3466 1997", April 1997, RFC 2130.
- 3467 [SMIv2-TC] J. Case, et al. "Textual Conventions for Version 2 of the Simple Network  
3468 Management Protocol (SNMPv2)", RFC 1903, January 1996.
- 3469 [tipsi] IEEE 1284.1, Transport-independent Printer System Interface (TIPSI).
- 3470 [URI-spec] Berners-Lee, T., Masinter, L., McCahill, M. , "Uniform Resource Locators  
3471 (URL)", RFC 1738, December, 1994.
- 3472 [US-ASCII] Coded Character Set - 7-bit American Standard Code for Information  
3473 Interchange, ANSI X3.4-1986.

3474 | [UTF-8] F. Yergeau, "UTF-8, a transformation format of Unicode and ISO 10646", RFC  
3475 | 2044, October 1996.

3476 | **8. Author's Addresses**

3477 | Ron Bergman  
3478 | Dataproducts Corp.  
3479 | 1757 Tapo Canyon Road  
3480 | Simi Valley, CA 93063-3394

3481 |  
3482 | Phone: 805-578-4421  
3483 | Fax: 805-578-4001  
3484 | Email: rbergman@dpc.com  
3485 |

3486 |  
3487 | Tom Hastings  
3488 | Xerox Corporation, ESAE-231  
3489 | 701 S. Aviation Blvd.  
3490 | El Segundo, CA 90245  
3491 |  
3492 | Phone: 310-333-6413  
3493 | Fax: 310-333-5514  
3494 | EMail: hastings@cp10.es.xerox.com  
3495 |

3496 |  
3497 | Scott A. Isaacson  
3498 | Novell, Inc.  
3499 | 122 E 1700 S  
3500 | Provo, UT 84606  
3501 |  
3502 | Phone: 801-861-7366  
3503 | Fax: 801-861-4025  
3504 | EMail: scott\_isaacson@novell.com  
3505 |

3506 |  
3507 | Harry Lewis  
3508 | IBM Corporation  
3509 | 6300 Diagonal Hwy  
3510 | Boulder, CO 80301  
3511 |  
3512 | Phone: (303) 924-5337

- 3513 Fax:  
3514 Email: [harryl@us.ibm.com](mailto:harryl@us.ibm.com)  
3515  
3516  
3517 Send comments to the printmib WG using the Job Monitoring Project (JMP)  
3518 Mailing List: [jmp@pwg.org](mailto:jmp@pwg.org)  
3519  
3520 To learn how to subscribe, send email to: [jmp-request@pwg.org](mailto:jmp-request@pwg.org)  
3521  
3522 For further information, access the PWG web page under "JMP":  
3523 <http://www.pwg.org/>  
3524
- 3525 Other Participants:
- 3526 Chuck Adams - Tektronix  
3527 Jeff Barnett - IBM  
3528 Keith Carter, IBM Corporation  
3529 Jeff Copeland - QMS  
3530 Andy Davidson - Tektronix  
3531 Roger deBry - IBM  
3532 Mabry Dozier - QMS  
3533 Lee Ferrel - Canon  
3534 Steve Gebert - IBM  
3535 Robert Herriot - Sun Microsystems Inc.  
3536 Shige Kanemitsu - Kyocera  
3537 David Kellerman - Northlake Software  
3538 Rick Landau - Digital  
3539 Harry Lewis - IBM  
3540 Pete Loya - HP  
3541 Ray Lutz - Cognisys  
3542 Jay Martin - Underscore  
3543 Mike MacKay, Novell, Inc.  
3544 Stan McConnell - Xerox  
3545 Carl-Uno Manros, Xerox, Corp.  
3546 Pat Nogay - IBM  
3547 Bob Pentecost - HP  
3548 Rob Rhoads - Intel  
3549 David Roach - Unisys  
3550 Hiroyuki Sato - Canon  
3551 Bob Setterbo - Adobe  
3552 Gail Songer, EFI

3553 Mike Timperman - Lexmark  
3554 Randy Turner - Sharp  
3555 William Wagner - Digital Products  
3556 Jim Walker - Dazel  
3557 Chris Wellens - Interworking Labs  
3558 Rob Whittle - Novell  
3559 Don Wright - Lexmark  
3560 Lloyd Young - Lexmark  
3561 Atsushi Yuki - Kyocera  
3562 Peter Zehler, Xerox, Corp.

3563 9. INDEX

3564 This index includes the textual conventions, the objects, and the attributes. Textual  
 3565 conventions all start with the prefix: "JM" and end with the suffix: "TC". Objects all  
 3566 starts with the prefix: "jm" followed by the group name. Attributes are identified with  
 3567 enums, and so start with any lower case letter and have no special prefix.

		3601	jmGeneralNewestActiveJobIndex .....	75	
3568	—C—	3602	jmGeneralNumberOfActiveJobs .....	74	
		3603	jmGeneralOldestActiveJobIndex .....	74	
3569	colorantConsumed .....	58	3604	jmJobIDJobIndex .....	78
3570	colorantRequested .....	58	3605	jmJobIDJobSetIndex .....	78
		3606	jmJobImpressionsCompleted .....	82	
3571	—D—	3607	jmJobImpressionsRequested .....	82	
		3608	jmJobIndex .....	80	
3572	deviceNameRequested .....	49	3609	jmJobKOctetsProcessed .....	82
3573	documentCopiesCompleted .....	55	3610	jmJobKOctetsRequested .....	81
3574	documentCopiesRequested .....	54	3611	jmJobOwner .....	83
3575	documentFormat .....	51	3612	JmJobServiceTypesTC .....	62
3576	documentFormatIndex .....	51	3613	JmJobSourcePlatformTypeTC .....	34
3577	documentName .....	50	3614	jmJobState .....	80
		3615	jmJobStateReasons1 .....	80	
3578	—F—	3616	JmJobStateReasons1TC .....	63	
		3617	JmJobStateReasons2TC .....	67	
3579	fileName .....	50	3618	JmJobStateReasons3TC .....	71
3580	finishing .....	53	3619	JmJobStateReasons4TC .....	72
3581	fullColorImpressionsCompleted .....	56	3620	JmJobStateTC .....	42
		3621	JmJobStringTC .....	33	
3582	—H—	3622	jmJobSubmissionID .....	77	
		3623	JmJobSubmissionTypeTC .....	40	
3583	highlightColorImpressionsCompleted .....	56	3624	JmMediumTypeTC .....	38
		3625	jmNumberOfInterveningJobs .....	81	
3584	—I—	3626	JmPrinterResolutionTC .....	37	
		3627	JmPrintQualityTC .....	37	
3585	impressionsCompletedCurrentCopy .....	56	3628	JmTimeStampTC .....	34
3586	impressionsInterpreted .....	56	3629	JmTonerEconomyTC .....	38
3587	impressionsSentToDevice .....	56	3630	JmUTF8StringTC .....	33
3588	impressionsSpooled .....	56	3631	jobAccountName .....	47
		3632	jobCodedCharSet .....	46	
3589	—J—	3633	jobComment .....	51	
		3634	jobCompletedTime .....	60	
3590	jmAttributeInstanceIndex .....	85	3635	jobCopiesCompleted .....	54
3591	jmAttributeTypeIndex .....	85	3636	jobCopiesRequested .....	54
3592	JmAttributeTypeTC .....	45	3637	jobHold .....	52
3593	jmAttributeValueAsInteger .....	86	3638	jobHoldUntil .....	52
3594	jmAttributeValueAsOctets .....	87	3639	jobKOctetsTransferred .....	55
3595	JmBooleanTC .....	38	3640	jobName .....	48
3596	JmFinishingTC .....	35	3641	jobOriginatingHost .....	49
3597	jmGeneralAttributePersistence .....	75	3642	jobPriority .....	52
3598	jmGeneralJobPersistence .....	75	3643	jobProcessAfterDateAndTime .....	52
3599	jmGeneralJobSetIndex .....	73	3644	jobProcessingCPUTime .....	60
3600	jmGeneralJobSetName .....	76	3645	jobServiceTypes .....	48



3646	jobSourceChannelIndex .....	49	3666	pagesRequested .....	57
3647	jobSourcePlatformType.....	49	3667	physicalDevice .....	50
3648	jobStartedBeingHeldTime.....	59	3668	printerResolutionRequested .....	53
3649	jobStartedProcessingTime .....	60	3669	printerResolutionUsed .....	54
3650	jobStateReasons2.....	46	3670	printQualityRequested.....	53
3651	jobStateReasons3.....	46	3671	printQualityUsed .....	53
3652	jobStateReasons4.....	46	3672	processingMessage .....	46
3653	jobSubmissionTime .....	59			
3654	jobSubmissionToServerTime .....	59	3673	—Q—	
3655	—M—		3674	queueNameRequested .....	50
3656	mediumConsumed .....	58	3675	—S—	
3657	mediumRequested .....	58			
			3676	serverAssignedJobName .....	47
3658	—N—		3677	sheetsCompleted.....	57
			3678	sheetsCompletedCurrentCopy .....	57
3659	numberOfDocuments .....	50	3679	sheetsRequested .....	57
			3680	sides .....	53
3660	—O—		3681	submittingApplicationName .....	49
			3682	submittingServerName .....	49
3661	other.....	46			
3662	outputBin .....	53	3683	—T—	
3663	—P—		3684	tonerDensityRequested .....	54
			3685	tonerDensityUsed .....	54
3664	pagesCompleted.....	57	3686	tonerEcomonyRequested.....	54
3665	pagesCompletedCurrentCopy .....	57	3687	tonerEcomonyUsed.....	54
3688					