

# 1. Job Monitoring MIB, V0.843

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

From: Tom Hastings

Date: 07/21/97

Version: 0.843

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

Status: ~~Seventh~~<sup>Sixth</sup> draft MIB that corresponds to editorial comments on V0.83 and changes to keep in alignment with IPP (printer-resolution syntax). ~~the changes agreed to at the JMP meeting, on Friday, 6/27/97. The major changes were to move the **jobOwner** attribute to the **jmJobTable**, so that no attributes are MANDATORY. However, we agreed to restore Ron's requirement that an attribute SHALL be implemented, if the server or device implements the corresponding functionality and it is available to the agent. We also deleted the **deviceAlertCode** attribute since it is in the Printer MIB. We deleted the **timeSinceXxxx** attributes since they can be computed from other attributes.~~

~~We agreed to make the random number and sequential numbers in the **jmJobSubmissionID** be last, so that a partial ID could be specified in a GetNext and step through all jobs with the same more significant part of **jmJobSubmissionID**. Harry and I had an action item about the use of IMPLIED and its interaction with such a specification. We have agreed that making **jmJobSubmissionID** fixed length with trailing spaces before the 8 digit number works with V1 and V2, since no length tag shall be present for fixed length. 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~~Three~~ 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-mibv.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.



36 INTERNET-DRAFT

37

38

39

40

41

42

43

44

45

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

46

**Job Monitoring MIB - V0.843**

47

**<draft-ietf-printmib-job-monitor-042.txt>**

48

**Expires Jan 2114, 1997**

49

**50 Status of this Memo**

51

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.

52

53

54

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."

55

56

57

58

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).

59

60

61

62

**Abstract**

63

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.

64

65

66

67

68

69

70

71

72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100  
101  
102  
103  
104  
105

**TABLE OF CONTENTS**

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

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

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

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

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

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

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

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

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

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

        3.1.1 Conformance Terminology .....18

        3.1.2 Agent Conformance Requirements .....18

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

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

            3.1.2.3 Printer MIB objects .....19

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

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

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

        3.3.1 Conformance of Attribute Implementation.....22

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

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

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

        3.3.5 Requested Attributes.....24

        3.3.6 Consumption Attributes.....25

        3.3.7 Index Value Attributes .....25

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

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

**3.6 IANA Considerations.....26**

        3.6.1 IANA Registration of enums .....26

            3.6.1.1 Type 1 enumerations.....27

            3.6.1.2 Type 2 enumerations.....27

            3.6.1.3 Type 3 enumeration .....28

106	3.6.2 IANA Registration of type 2 bit values .....	28
107	3.6.3 IANA Registration of Job Submission Id Formats.....	28
108	3.6.4 IANA Registration of MIME types/sub-types for document-formats .....	28
109	<b>3.7 Security Considerations .....</b>	<b>29</b>
110	3.7.1 Read-Write objects .....	29
111	3.7.2 Read-Only Objects In Other User's Jobs.....	29
112	<b>3.8 Values for Objects.....</b>	<b>22</b>
113	<b>3.9 Notification-s .....</b>	<b>29</b>
114	<b>4. MIB SPECIFICATION .....</b>	<b>29</b>
115	<b>Textual conventions for this MIB module .....</b>	<b>31</b>
116	JmTimeStampTC - simple time in seconds.....	31
117	JmJobSourcePlatformTypeTC - operating system platform definitions.....	31
118	JmFinishingTC - device finishing definitions.....	32
119	JmPrintQualityTC - print quality.....	33
120	JmPrinterResolutionTC - printer resolution.....	34
121	JmTonerEconomyTC - toner economy setting.....	35
122	JmBooleanTC - Boolean value .....	36
123	JmMediumTypeTC - medium type definitions.....	36
124	JmJobSubmissionIDTypeTC - job submission ID type definitions.....	37
125	JmJobStateTC - job state definitions.....	39
126	JmAttributeTypeTC - attribute type definitions.....	42
127	other (Int32(-2..) and/or Octets63) .....	42
128	Job State attributes.....	43
129	jobStateReasons2 (JmJobStateReasons2TC) .....	43
130	jobStateReasons3 (JmJobStateReasons3TC) .....	43
131	jobStateReasons4 (JmJobStateReasons4TC) .....	43
132	processingMessage (Octets63).....	43
133	Job Identification attributes.....	43
134	jobAccountName (Octets63).....	43
135	serverAssignedJobName (Octets63) .....	43
136	jobName (Octets63).....	44
137	jobServiceTypes (JmJobServiceTypesTC).....	44
138	jobSourceChannelIndex (Int32(0..)).....	45
139	jobSourcePlatformType (JmJobSourcePlatformTypeTC).....	45
140	submittingServerName (Octets63).....	45
141	submittingApplicationName (Octets63) .....	45
142	jobOriginatingHost (Octets63) .....	45
143	deviceNameRequested (Octets63).....	45
144	queueNameRequested (Octets63) .....	45
145	physicalDevice (hrDeviceIndex and/or Octets63) .....	45
146	numberOfDocuments (Int32(-2..)).....	46
147	fileName (Octets63).....	46
148	documentName (Octets63).....	46

149	jobComment (Octets63) .....	46
150	documentFormatIndex (Int32(0..)) .....	46
151	documentFormat (PrtInterpreterLangFamilyTC and/or Octets63).....	46
152	Job Parameter attributes.....	47
153	jobPriority (Int32(1..100)).....	47
154	jobProcessAfterDateAndTime (DateAndTime).....	47
155	jobHold (JmBooleanTC).....	47
156	jobHoldUntil (Octets63) .....	47
157	outputBin (Int32(0..) and/or Octets63) .....	48
158	sides (Int32(-2..2)) .....	48
159	finishing (JmFinishingTC).....	48
160	Image Quality attributes (requested and used).....	48
161	printQualityRequested (JmPrintQualityTC).....	48
162	printQualityUsed (JmPrintQualityTC).....	48
163	printerResolutionRequested (JmPrinterResolutionTC).....	48
164	printerResolutionUsed (JmPrinterResolutionTC).....	48
165	tonerEcomonyRequested (JmTonerEconomyTC).....	48
166	tonerEcomonyUsed (JmTonerEconomyTC).....	48
167	tonerDensityRequested (Int32(-2..100)).....	49
168	tonerDensityUsed (Int32(-2..100)).....	49
169	Job Progress attributes (requested and consumed) .....	49
170	jobCopiesRequested (Int32(-2..)).....	49
171	jobCopiesCompleted (Int32(-2..)).....	49
172	documentCopiesRequested (Int32(-2..)).....	49
173	documentCopiesCompleted (Int32(-2..)).....	49
174	jobKOctetsTransferred (Int32(-2..)).....	49
175	Impression attributes (requested and consumed) .....	50
176	impressionsSpooled (Int32(-2..)).....	50
177	impressionsSentToDevice (Int32(-2..)).....	50
178	impressionsInterpreted (Int32(-2..)) .....	50
179	impressionsCompletedCurrentCopy (Int32(-2..)).....	50
180	fullColorImpressionsCompleted (Int32(-2..)).....	50
181	highlightColorImpressionsCompleted (Int32(-2..)).....	51
182	Page attributes (requested and consumed).....	51
183	pagesRequested (Int32(-2..)) .....	51
184	pagesCompleted (Int32(-2..)) .....	51
185	pagesCompletedCurrentCopy (Int32(-2..)) .....	51
186	Sheet attributes (requested and consumed).....	51
187	sheetsRequested (Int32(-2..)).....	51
188	sheetsCompleted (Int32(-2..)).....	51
189	sheetsCompletedCurrentCopy (Int32(-2..)).....	51
190	Resource attributes (requested and consumed) .....	52
191	mediumRequested (JmMediumTypeTC and/or Octets63) .....	52
192	mediumConsumed (Octets63) .....	52
193	colorantRequested (Int32(-2..) and/or Octets63) .....	52
194	colorantConsumed (Int32(-2..) and/or Octets63) .....	52
195	Time attributes (set by server or device).....	52
196	jobSubmissionToServerTime (JmTimeStampTC and/or DateAndTime).....	53
197	jobSubmissionTime (JmTimeStampTC and/or DateAndTime).....	53

198	jobStartedBeingHeldTime (JmTimeStampTC and/or DateAndTime) .....	53
199	jobStartedProcessingTime (JmTimeStampTC and/or DateAndTime) .....	53
200	jobCompletedTime (JmTimeStampTC and/or DateAndTime) .....	54
201	jobProcessingCPUTime (Int32(-2..)) .....	54
202	JmJobServiceTypesTC - bit encoded job service type definitions.....	56
203	JmJobStateReasons1TC - additional information about job states .....	57
204	JmJobStateReasons2TC - More additional information about job states .....	60
205	JmJobStateReasons3TC - More additional information about job states .....	63
206	JmJobStateReasons4TC - More additional information about job states .....	63
207	<b>The General Group (MANDATORY) .....</b>	<b>65</b>
208	jmGeneralJobSetIndex (Int32(1..32767)).....	65
209	jmGeneralNumberOfActiveJobs (Int32(0..)).....	66
210	jmGeneralOldestActiveJobIndex (Int32(0..)).....	66
211	jmGeneralNewestActiveJobIndex (Int32(0..)).....	66
212	jmGeneralJobPersistence (Int32(15..)).....	67
213	jmGeneralAttributePersistence (Int32(15..)).....	67
214	jmGeneralJobSetName (Octets63).....	67
215	<b>The Job ID Group (MANDATORY).....</b>	<b>68</b>
216	jmJobSubmissionID (OCTET STRING(SIZE(48))).....	69
217	jmJobIDJobSetIndex (Int32(1..32767)).....	69
218	jmJobIDJobIndex (Int32(1..)).....	69
219	<b>The Job Group (MANDATORY).....</b>	<b>70</b>
220	jmJobIndex (Int32(1..)).....	71
221	jmJobState (JmJobStateTC).....	71
222	jmJobStateReasons1 (JmJobStateReasons1TC).....	71
223	jmNumberOfInterveningJobs (Int32(-2..)).....	72
224	jmJobKOctetsRequested (Int32(-2..)).....	72
225	jmJobKOctetsProcessed (Int32(-2..)).....	72
226	jmJobImpressionsRequested (Int32(-2..)).....	73
227	jmJobImpressionsCompleted (Int32(-2..)).....	73
228	jmJobOwner (Octets63).....	73
229	<b>The Attribute Group (MANDATORY) .....</b>	<b>74</b>
230	jmAttributeTypeIndex (JmAttributeTypeTC).....	75
231	jmAttributeInstanceIndex (Int32(1..32767)).....	75
232	jmAttributeValueAsInteger (Int32(-2..)).....	76
233	jmAttributeValueAsOctets (Octets63) .....	76
234	<b>5. APPENDIX A - IMPLEMENTING THE JOB LIFE CYCLE.....</b>	<b>80</b>
235	<b>6. APPENDIX B - SUPPORT OF THE JOB SUBMISSION ID IN JOB</b>	
236	<b>SUBMISSION PROTOCOLS .....</b>	<b>80</b>
237	<b>6.1 Hewlett-Packard's Printer Job Language (PJJ) .....</b>	<b>81</b>

238      **6.2 ISO DPA.....81**

239      **7. REFERENCES..... 81**

240      **8. AUTHOR'S ADDRESSES..... 82**

241      **9. INDEX ..... 85**

242



243

## Job Monitoring MIB

### 244 1. Introduction

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

254 The Job Monitoring MIB consists of a ~~7-object~~ General Group, a ~~2-object~~ Job Submission  
255 ID Group, a ~~7-object~~ Job Group, and an ~~2-object~~ Attribute Group. Each group is a table.  
256 All accessible objects are read-only. The General Group contains general information that  
257 applies to all jobs in a job set. The Job Submission ID table maps the job submission ID  
258 that the client uses to identify a job to the **jmJobIndex** that the Job Monitoring Agent  
259 uses to identify jobs in the Job and Attribute tables. The Job table contains the  
260 MANDATORY integer job state and status objects. The Attribute table consists of  
261 multiple entries per job that specify (1) job and document identification and parameters,  
262 (2) requested resources, and (3) consumed resources during and after job  
263 processing/printing. Sixty five job attributes are defined as textual conventions that an  
264 agent SHALL return if the server or device implements the functionality so represented  
265 and the agent has access to the information.

#### 266 1.1 Types of Information in the MIB

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

269 User:

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

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

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

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

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

286 Capacity Planner:  
287 Provide the ability to determine printer utilization as a function of time.  
288 Provide the ability to determine how long jobs wait before starting to print.

289 Accountant:  
290 Provide information to allow the creation of a record of resources consumed and  
291 printer usage data for charging users or groups for resources consumed.  
292 Provide information to allow the prediction of consumable usage and resource  
293 need.

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

## 299 **1.2 Types of Job Monitoring Applications**

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

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

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

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

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

## 334 2. Terminology and Job Model

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

389 User: ~~is~~-a person that uses a client or a monitor.

390 End User: ~~is~~-a user that uses a client to submit a print job.

391 System Operator: ~~is~~-a user that uses a monitor to monitor the system and carries out tasks  
392 to keep the system running.

393 System Administrator: ~~is~~-a user that specifies policy for the system.

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

398 Document Instruction: ~~is~~-an instruction specifying how to process the document.  
399 Document instructions MAY be passed in the job submission protocol separate from the  
400 actual document data, or MAY be embedded in the document data or a combination,  
401 depending on the job submission protocol and implementation.

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

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

414 Job Monitoring (using SNMP): ~~is~~the activity of a management application of accessing  
415 the MIB and (1) identifying jobs in the job tables~~within the serial streams of data~~ being  
416 processed by the server, printer or other devices, (2) ~~creating "rows" in the job table for~~  
417 ~~each job, and~~ (2) displaying~~recording~~ information to the user, known by the agent, about  
418 the processing of the job ~~in that "row"~~.

419 Job Accounting: ~~is the activity of a management application of accessing the MIB and~~  
 420 recording what happens to the job during and after the processing ~~and printing~~ of the job.

421 **2.1 System Configurations for the Job Monitoring MIB**

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

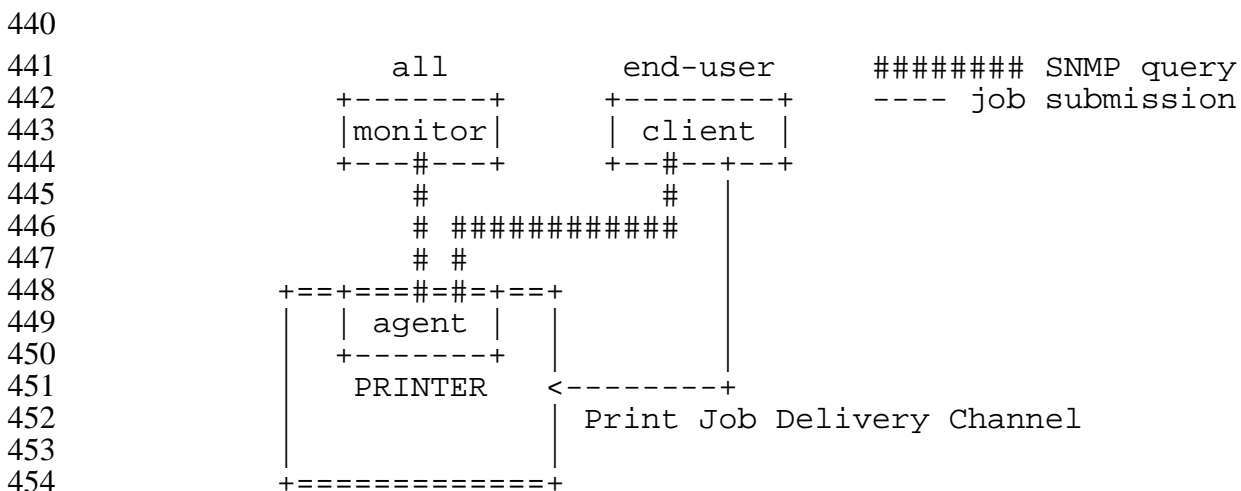
425 The diagram in the Printer MIB[print-mib] entitled: "One Printer's View of the Network"  
 426 is assumed for this MIB as well. Please refer to that diagram to aid in understanding the  
 427 following system configurations.

428 **2.1.1 Configuration 1 - client-printer**

429 In the **client-printer** configuration, the **client(s)** submit jobs directly to the printer, either  
 430 by some direct connect, or by network connection. ~~The client-printer configuration can~~  
 431 ~~accommodate multiple job submitting clients in either of two ways:~~

- 432 ~~1. if each client relinquishes control of the Print Job Delivery Channel after each~~  
 433 ~~job (or after a number of jobs)~~
- 434 ~~1. if the printer supports more than one Print Job Delivery Channel~~

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



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

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

- 458 1. Multiple **clients** MAY submit jobs to a **printer**.
- 459 2. Multiple **clients** MAY monitor a **printer**.
- 460 3. Multiple **monitors** MAY monitor a **printer**.
- 461 4. A **client** MAY submit jobs to multiple **printers**.
- 462 5. A **monitor** MAY monitor multiple **printers**.

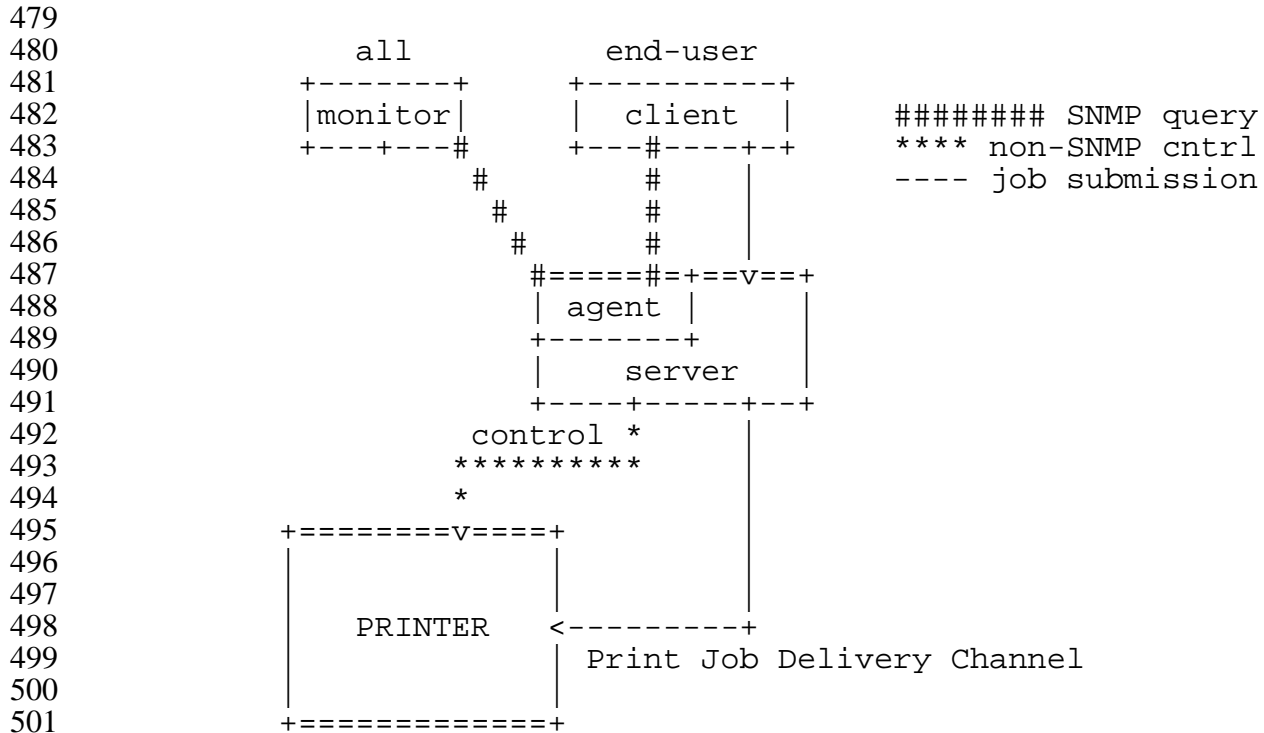
### 463 2.1.2 Configuration 2 - client-server-printer - agent in the server

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

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

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

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



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

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

- 505 1. Multiple **clients** MAY submit jobs to a **server**.
- 506 2. Multiple **clients** MAY monitor a **server**.
- 507 3. Multiple **monitors** MAY monitor a **server**.
- 508 4. A **client** MAY submit jobs to multiple **servers**.
- 509 5. A **monitor** MAY monitor multiple **servers**.
- 510 6. Multiple **servers** MAY submit jobs to a **printer**.
- 511 7. Multiple **servers** MAY control a **printer**.

512 **2.1.3 Configuration 3 - client-server-printer - client monitors printer agent and**  
 513 **server**

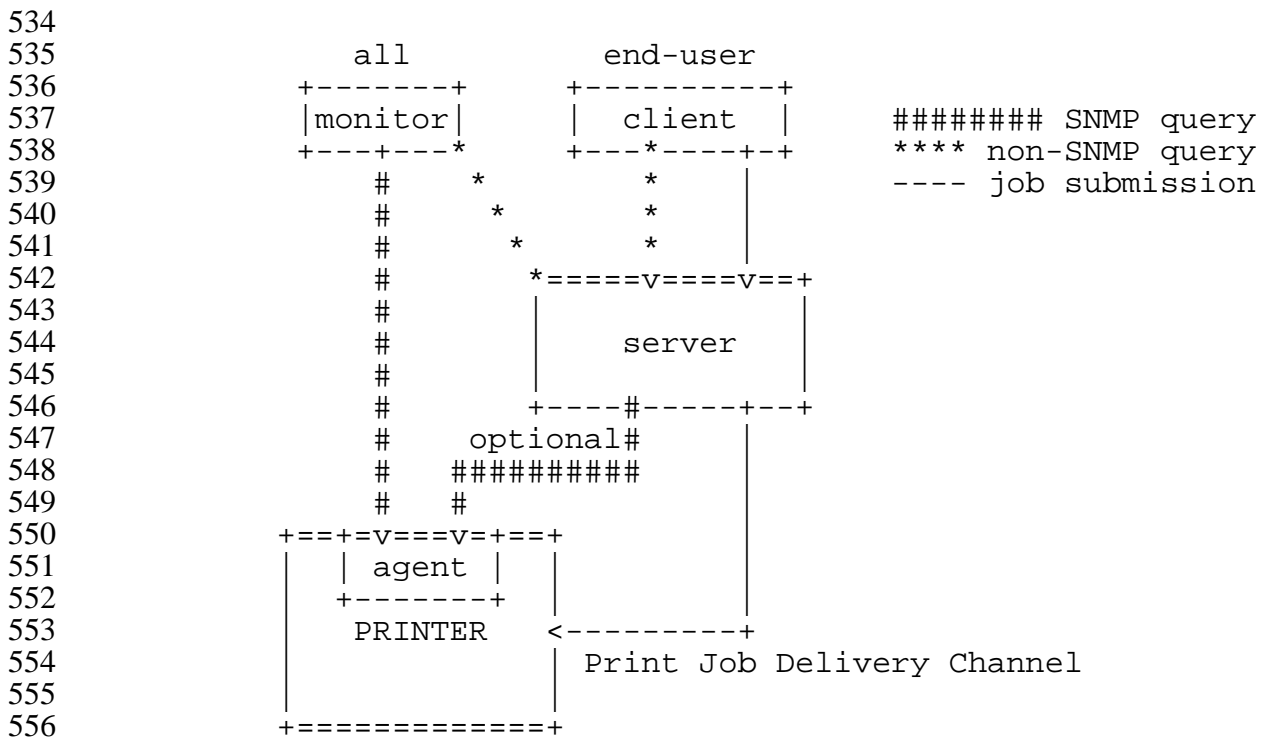
514 In the **client-server-printer** configuration 3, the **client(s)** submit jobs to an intermediate  
 515 **server** by some network connection, *not* directly to the **printer**. That server does not  
 516 contain a Job Monitoring MIB and agent.

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



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

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



557 **Figure 2-3 - Configuration 3 - client-server-printer - client monitors printer agent**  
 558 **and server**

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

- 561 1. Multiple **clients** MAY submit jobs to a **server**.
- 562 2. Multiple **clients** MAY monitor a **server**.
- 563 3. Multiple **monitors** MAY monitor a **server**.
- 564 4. A **client** MAY submit jobs to multiple **servers**.
- 565 5. A **monitor** MAY monitor multiple **servers**.
- 566 6. Multiple **servers** MAY submit jobs to a **printer**.
- 567 7. Multiple **servers** MAY control a **printer**.

### 568 3. Managed Object Usage

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

#### 570 3.1 Conformance Considerations

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

##### 574 3.1.1 Conformance Terminology

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

- 577 • "SHALL": indicates an action that the subject of the sentence must implement in  
578 order to claim conformance to this specification
- 579 • "MAY": indicates an action that the subject of the sentence does not have to  
580 implement in order to claim conformance to this specification, in other words that  
581 action is an implementation option
- 582 • "NEED NOT": indicates an action that the subject of the sentence does not have to  
583 implement in order to claim conformance to this specification. The verb "NEED  
584 NOT" is used instead of "may not", since "may not" sounds like a prohibition.
- 585 • "SHOULD": indicates an action that is recommended for the subject of the  
586 sentence to implement, but is not required, in order to claim conformance to this  
587 specification.

##### 588 3.1.2 Agent Conformance Requirements

589 A conforming agent:

- 590 1. SHALL implement *all* MANDATORY groups in this specification.

591 2. SHALL implement any attributes if (1) the server or device supports the functionality  
592 represented by the attribute and (2) the information is available to the agent.

593 3. SHOULD implement both forms of an attribute if it implements an attribute that  
594 permits a choice of INTEGER and OCTET STRING forms, since implementing both  
595 forms may help management applications by giving them a choice of representations,  
596 since the representation are equivalent. See the **JmAttributeTypeTC** textual-  
597 convention.

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

#### 600 3.1.2.1 MIB II System Group objects

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

#### 603 3.1.2.2 MIB II Interface Group objects

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

#### 606 3.1.2.3 Printer MIB objects

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

### 613 3.1.3 Job Monitoring Application Conformance Requirements

614 A conforming job monitoring application:

615 1. SHALL accept the full syntactic range for all objects in all MANDATORY groups and  
616 all MANDATORY attributes that are required to be implemented by an agent  
617 according to Section 3.1.2 and SHALL either present them to the user or ignore them.

618 2. SHALL accept the full syntactic range for *all* attributes, including enum and bit values  
619 specified in this specification and additional ones that may be registered with IANA  
620 and SHALL either present them to the user or ignore them. In particular, a  
621 conforming job monitoring application SHALL not malfunction when receiving any  
622 standard or registered enum or bit values. See Section 3.6 entitled "IANA  
623 Considerations".

- 624 3. SHALL NOT fail when operating with agents that materialize attributes *after* the job  
625 has been submitted, as opposed to when the job is submitted.
- 626 4. SHALL, if it supports a time attribute, accept either form of the time attribute, since  
627 agents are free to implement either time form.

### 628 3.2 The Job Tables and the Oldest Active and Newest Active Indexes

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

- 631 1. **jmGeneralJobSetIndex** - which job set  
632 2. **jmJobIndex** - which job in the job set

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

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

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

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

649 Each time a new job is accepted by the server or device that the agent is providing access  
650 to AND that job is to be 'active' (**pending**, **processing**, or **processingStopped**, but not  
651 **pendingHeld**), the agent SHALL copy the value of the job's **jmJobIndex** to the  
652 **jmGeneralNewestActiveJobIndex** object. If the new job is to be 'inactive'  
653 (**pendingHeld** state), the agent SHALL not change the value of  
654 **jmGeneralNewestActiveJobIndex** object.

655 When a job transitions from one of the 'active' job states (**pending**, **processing**,  
656 **processingStopped**) to one of the 'inactive' job states (**pendingHeld**, **completed**,  
657 **canceled**, or **aborted**), with a **jmJobIndex** value that matches the  
658 **jmGeneralOldestActiveJobIndex** object, the agent SHALL advance (or wrap) the value

659 to the next oldest 'active' job, if any. See the **JmJobStateTC** textual-convention for a  
660 definition of the job states.

661 Whenever a job ~~transition~~changes from one of the 'inactive' job states to one of the  
662 'active' job states (from **pendingHeld** to **pending** or **processing**), the agent SHALL  
663 update the value of either the **jmGeneralOldestActiveJobIndex** or the  
664 **jmGeneralNewestActiveJobIndex** objects, or both, if the job's **jmJobIndex** value is  
665 outside the range between **jmGeneralOldestActiveJobIndex** and  
666 **jmGeneralNewestActiveJobIndex**.

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

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

674 If an application detects that the **jmGeneralNewestActiveJobIndex** is smaller than  
675 **jmGeneralOldestActiveJobIndex**, the job index has wrapped. In this case, ~~when the~~  
676 ~~application detects that the returned OID is in a different MIB (Get Next has moved to the~~  
677 ~~next MIB in the agent),~~ the application SHALL reset the index to start over at 1 when the  
678 end of the table is reached and continue the GetNext operations to find the rest of the  
679 active jobs.

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

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

### 686 3.3 The Attribute Mechanism

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

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

694 1. **jmGeneralJobSetIndex** - which job set

- 695 2. **jmJobIndex** - which job in the job set  
696 3. **jmAttributeTypeIndex** - which attribute  
697 4. **jmAttributeInstanceIndex** - which attribute instance for those attributes that can  
698 have multiple values per job.

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

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

### 712 3.3.1 Conformance of Attribute Implementation

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

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

### 721 3.3.2 Useful, 'Unknown', and 'Other' Values for Objects and Attributes

722 Some attributes have a 'useful' ~~Integer32~~INTEGER32 value, some have a 'useful' OCTET  
723 STRING value, some MAY have either or both depending on implementation, and some  
724 MUST have both. See the **JmAttributeTypeTC** textual convention for the specification  
725 of each attribute.

726 SNMP requires that if an object cannot be implemented because its values cannot be  
727 accessed, then a compliant agent SHALL return an SNMP error in SNMPv1 or an  
728 exception value in SNMPv2. However, this MIB has been designed so that 'all' objects  
729 can and SHALL be implemented by an agent, so that neither the SNMPv1 error nor the

730 SNMPv2 exception value SHALL be generated by the agent. This MIB has also been  
 731 designed so that when an agent materializes an attribute, the agent SHALL materialize a  
 732 row consisting of both the **jmAttributeValueAsInteger** and **jmAttributeValueAsOctets**  
 733 objects.

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

739 Since each attribute is represented by a row consisting of both the  
 740 **jmAttributeValueAsInteger** and **jmAttributeValueAsOctets** MANDATORY objects,  
 741 SNMP requires that the agent SHALL always create an attribute row with both objects  
 742 specified. However, for most attributes the agent SHALL return a "useful" value for one  
 743 of the objects and SHALL return the 'other' value for the other object. For integer only  
 744 attributes, the agent SHALL always return a zero-length string value for the  
 745 **jmAttributeValueAsOctets** object. For octet string only attributes, the agent SHALL  
 746 always return a **'-1'** value for the **jmAttributeValueAsInteger** object.

### 747 3.3.3 Data Sub-types and Attribute Naming Conventions

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

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

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

'Octets(m..n)'

For all other OCTET STRING ranges, the exact range is indicated.

### 763 3.3.4 Single-Value (Row) Versus Multi-Value (MULTI-ROW) Attributes

764 Most attributes SHALL have only one row per job. However, a few attributes can have  
 765 multiple values per job or even per document, where each value is a separate row in the  
 766 **jmAttributeTable**. Unless indicated with '**MULTI-ROW:**' in the **JmAttributeTypeTC**  
 767 description, an agent SHALL ensure that each attribute occurs only once in the  
 768 **jmAttributeTable** for a job. Most of the '**MULTI-ROW**' attributes do not allow  
 769 duplicate values, i.e., the agent SHALL ensure that each value occurs only once for a job.  
 770 Only if the specification of the '**MULTI-ROW**' attribute also says "the values NEED NOT  
 771 be unique" can the agent allow duplicate values to occur for the job. ~~Attributes that are~~  
 772 ~~permitted to appear multiple times in the **jmAttributeTable** for a job are indicated with~~  
 773 ~~'**MULTI-ROW:**' in their specification in the **JmAttributeTypeTC**. However, such~~  
 774 ~~'**MULTI-ROW**' attributes SHALL *not* contain duplicates for 'intensive' (as opposed to~~  
 775 ~~'extensive') attributes.~~

776 NOTE - Duplicate are allowed for 'extensive' '**MULTI-ROW**' attributes, such as  
 777 **fileName(34)** or **documentName(35)**, but are not allowed for 'intensive' '**MULTI-ROW**'  
 778 attributes, such as **mediumConsumed(171)** and **documentFormat(38)**. ~~For example, a~~  
 779 ~~job or document(s) may use multiple PDLs. However, each distinct **documentFormat**~~  
 780 ~~attribute value SHALL appear in the **jmAttributeTable** only once for a job since the~~  
 781 ~~interpreter language is an intensive attribute, even though the job has a number of~~  
 782 ~~documents that all use the same PDL.~~

783 ~~As another example of an intensive attribute that can have multiple entries, if a document~~  
 784 ~~or job uses multiple types of media, there SHALL be only one row in the~~  
 785 ~~**jmAttributeTable** for each media type, not one row for each document that uses that~~  
 786 ~~medium type.~~

787 ~~On the other hand, if a job contains two documents of the same name, there can be~~  
 788 ~~separate rows for the **documentName** attribute with the same name, since a document~~  
 789 ~~name is an extensive attribute. The specification indicates that the values NEED NOT be~~  
 790 ~~unique for such '**MULTI-ROW:** attributes'~~

### 791 3.3.5 Requested Attributes

792 A number of attributes record requirements for the job. Such attribute names end with the  
 793 word '**Requested**'. In the interests of brevity, the phrase 'requested' SHALL mean: (1)  
 794 requested by the client (or intervening server) in the job submission protocol ~~that~~  
 795 ~~submitted the job~~ and MAY also mean (2) embedded in the submitted document data,



796 and/or (3) defaulted by the recipient device or server with the same semantics as if the  
797 requester had supplied, depending on implementation.

### 798 3.3.6 Consumption Attributes

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

### 805 3.3.7 Index Value Attributes

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

## 812 3.4 Job Identification

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

824 The server/device-side identifier, called the **jmJobIndex** object, SHALL be assigned by  
825 the SNMP Job Monitoring MIB agent when the server or device accepts the jobs from  
826 submitting clients. The **jmJobIndex** object allows the interested party to obtain all  
827 objects desired that relate to this job. The MIB provides a mapping table that maps each  
828 Job Submission ID (generated by the client) to the corresponding **jmJobIndex** value  
829 generated by the agent, so that an application can determine the correct value for the

830 **jmJobIndex** value for the job of interest in a single Get operation, given the Job  
831 Submission ID. See the **jmJobIDGroup**.

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

### 835 3.5 Internationalization Considerations

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

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

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

### 851 3.6 IANA Considerations

852 During the development of this standard, the Printer Working Group (PWG) working with  
853 IANA [[iana](#)] will register additional enums while the standard is in the proposed and draft  
854 states according to the procedures described in this section. IANA will handle registration  
855 of additional enums after this standard is approved in cooperation with an IANA-  
856 appointed registration editor from the PWG according to the procedures described in this  
857 section:

#### 858 3.6.1 IANA Registration of enums

859 This specification uses textual conventions to define enumerated values (enums) and bit  
860 values. Enumerations (enums) and bit values are sets of symbolic values defined for use  
861 with one or more objects or attributes. All enumeration sets and bit value sets are  
862 assigned a symbolic data type name (textual convention). As a convention the symbolic

863 name ends in "TC" for textual convention. These enumerations are defined at the  
864 beginning of the MIB module specification.

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

#### 870 3.6.1.1 Type 1 enumerations

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

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

#### 874 3.6.1.2 Type 2 enumerations

875 Type 2 enumeration: An initial set of values are defined in the Job Monitoring MIB  
876 specification. Additional enumerated values are registered after review by this working  
877 group or an editor appointed by IANA after this working group is no longer active. ~~The~~  
878 ~~initial versions of the MIB will contain the values registered so far. After the MIB is~~  
879 ~~approved, additional values will be registered through IANA after approval by this~~  
880 ~~working group.~~

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

- 882 1. **JmTimeStampTC**
- 883 2. **JmFinishingTC** [same enum values as IPP "finishing" attribute]
- 884 3. **JmPrintQualityTC** [same enum values as IPP "print-quality" attribute]
- 885 4. **JmTonerEconomyTC**
- 886 ~~5. **JmPrinterResolutionTC** [same enum values as IPP "printer-resolution" attribute]~~
- 887 5. **JmMediumTypeTC**
- 888 6. **JmJobSubmissionTypeTC**
- 889 7. **JmJobStateTC** [same enum values as IPP "job-state" attribute]
- 890 8. **JmAttributeTypeTC**

891 ~~For those textual conventions that are labeled "[same enum values as IPP]" have the~~  
892 ~~same enum values as the indicated IPP Job attribute. When IPP registers additional~~  
893 ~~values, those values SHALL be simultaneously registered by IANA for use with the Job~~  
894 ~~Monitoring MIB textual convention, so that the enum values stay in lock step between the~~  
895 ~~IPP [ipp-model] protocol and the Job Monitoring MIB.~~

## 896 3.6.1.3 Type 3 enumeration

897 Type 3 enumeration: An initial set of values are defined in the Job Monitoring MIB  
898 specification. Additional enumerated values are registered through IANA without  
899 working group review. ~~The initial versions of the MIB will contain the values registered~~  
900 ~~so far. After the MIB is approved, additional values will be registered through IANA~~  
901 ~~without approval by this working group.~~

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

903 **3.6.2 IANA Registration of type 2 bit values**

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

- 905 1. **JmJobServiceTypesTC**
- 906 2. **JmJobStateReasons1TC**
- 907 3. **JmJobStateReasons2TC**
- 908 4. **JmJobStateReasons3TC**
- 909 5. **JmJobStateReasons4TC**

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

913 The registration of **JmJobServiceTypesTC** and **JmJobStateReasonsN**TC bit values  
914 SHALL follow the procedures for a type 2 enum as specified in Section 3.6.1.2.

915 **3.6.3 IANA Registration of Job Submission Id Formats**

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

920 **3.6.4 IANA Registration of MIME types/sub-types for document-formats**

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

925 **3.7 Security Considerations**

926 **3.7.1 Read-Write objects**

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

932 **3.7.2 Read-Only Objects In Other User's Jobs**

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

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

944 **3.8 Notifications**

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

949 **4. MIB specification**

950 The following pages constitute the actual Job Monitoring MIB.

```

951 Job-Monitoring-MIB DEFINITIONS ::= BEGIN
952
953 IMPORTS
    MODULE-IDENTITY, OBJECT-TYPE, experimental, Integer32
    TEXTUAL-CONVENTION
    MODULE-COMPLIANCE, OBJECT-GROUP
    FROM SNMPv2-SMI
    FROM SNMPv2-TC
    FROM SNMPv2-CONF;

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

954 -- Use the experimental (54) OID assigned to the Printer MIB[print-mib]
955 -- before it was published as RFC 1759.
956 -- Upon publication of the Job Monitoring MIB as an RFC, delete this
957 -- comment and the line following this comment and change the
958 -- reference of { temp 105 } (below) to { mib-2 X }.
959 -- This will result in changing:
960 -- 1 3 6 1 3 54 jobmonMIB(105)  to:
961 -- 1 3 6 1 2 1 jobmonMIB(X)
962 -- This will make it easier to translate prototypes to
963 -- the standard namespace because the lengths of the OIDs won't
964 -- change.
965 temp OBJECT IDENTIFIER ::= { experimental 54 }
966
967 jobmonMIB MODULE-IDENTITY
968     LAST-UPDATED "970721+40000Z"
969     ORGANIZATION "IETF Printer MIB Working Group"
970     CONTACT-INFO
971         "Tom Hastings
972         Postal: Xerox Corp.
973             Mail stop ESAE-231
974             701 S. Aviation Blvd.
975             El Segundo, CA 90245
976
977         Tel: (301)333-6413
978         Fax: (301)333-5514
979         E-mail: hstings@cp10.es.xerox.com
980
981         Send comments to the printmib WG using the Job Monitoring
982         Project (JMP) Mailing List: jmp@pwg.org
983
984         To learn how to subscribe to the JMP mailing list,
985         send email to: jmp-request@pwg.org
986
987         For further information, access the PWG web page under 'JMP':
988

```

```

989         http://www.pwg.org/"
990     DESCRIPTION
991         "The MIB module for monitoring job in servers, printers, and other devices.
992
993         File: draft-ietf-printmib-job-monitor-042.txt
994         Version: 0.843"
995     ::= { temp 105 }
996
997
998
999 -- Textual conventions for this MIB module
1000
1001
1002 JmTimeStampTC ::= TEXTUAL-CONVENTION
1003     STATUS      current
1004     DESCRIPTION
1005         "The simple time at which an event took place.  The units SHALL be in seconds since the
1006         system was booted.
1007
1008         NOTE - JmTimeStampTC is defined in units of seconds, rather than 100ths of seconds, so as
1009         to be simpler for agents to implement (even if they have to implement the 100ths of a second to
1010         comply with implementing sysUpTime in MIB-II[mib-II].)
1011
1012         NOTE - JmTimeStampTC is defined as an Integer32 so that it can be used as a value of an
1013         attribute, i.e., as a value of the jmAttributeValueAsInteger object.  The TimeStamp textual-
1014         convention defined in SMNPv2-TC is defined as an APPLICATION 3 IMPLICIT INTEGER
1015         tag, not an Integer32, so cannot be used in this MIB as one of the values of
1016         jmAttributeValueAsInteger."
1017     SYNTAX      INTEGER(0..2147483647)
1018
1019
1020
1021
1022 JmJobSourcePlatformTypeTC ::= TEXTUAL-CONVENTION
1023     STATUS      current
1024     DESCRIPTION
1025         "The source platform type that can submit jobs to servers or devices in any of the 3
1026         configurations."
1027     REFERENCE
1028         "This is a type 2 enumeration.  See Section 3.6.1.2."
1029     SYNTAX      INTEGER {
1030         other(1),
1031         unknown(2),
1032         sptUNIX(3),           -- UNIX(tm)
1033         sptOS2(4),         -- OS/2
1034         sptPCDOS(5),      -- DOS
1035         sptNT(6),         -- NT
    
```

```

1030     }
1031
1032
1033
1034
1035
1036 JmFinishingTC ::= TEXTUAL-CONVENTION
1037     STATUS      current
1038     DESCRIPTION
1039         "The type of finishing operation.
1040
1041         These values are the same as the enum values of the IPP 'finishings' attribute. See Section
1042         3.6.1.2.
1043
1044         other(1),
1045             Some other finishing operation besides one of the specified or registered values.
1046
1047         unknown(2),
1048             The finishing is unknown.
1049
1050         none(3),
1051             Perform no finishing.
1052
1053         staple(4),
1054             Bind the document(s) with one or more staples. The exact number and placement of the
1055             staples is site-defined.
1056
1057         stapleTopLeft(5),
1058             Place one or more staples on the top left corner of the document(s).
1059
1060         stapleBottomLeft(6),
1061             Place one or more staples on the bottom left corner of the document(s).
1062
1063         stapleTopRight(7),
1064             Place one or more staples on the top right corner of the document(s).
1065
1066         stapleBottomRight(8),
1067             Place one or more staples on the bottom right corner of the document(s).
1068
1069         saddleStitch(9),
1070             Bind the document(s) with one or more staples (wire stitches) along the middle fold. The
1071             exact number and placement of the stitches is site-defined.

```



1072  
1073 **edgeStitch(10),**  
1074 Bind the document(s) with one or more staples (wire stitches) along one edge. The exact  
1075 number and placement of the staples is site-defined.  
1076  
1077 **punch(11),**  
1078 This value indicates that holes are required in the finished document. The exact number  
1079 and placement of the holes is site-defined. The punch specification MAY be satisfied (in a  
1080 site- and implementation-specific manner) either by drilling/punching, or by substituting  
1081 pre-drilled media.  
1082  
1083 **cover(12),**  
1084 This value is specified when it is desired to select a non-printed (or pre-printed) cover for  
1085 the document. This does not supplant the specification of a printed cover (on cover stock  
1086 medium) by the document itself.  
1087  
1088 **bind(13)**  
1089 This value indicates that a binding is to be applied to the document; the type and  
1090 placement of the binding is product-specific."  
1091 REFERENCE  
1092 "This is a type 2 enumeration. See Section 3.6.1.2."  
1093 SYNTAX INTEGER {  
1094 other(1),  
1095 unknown(2),  
1096 none(3),  
1097 staple(4),  
1098 stapleTopLeft(5),  
1099 stapleBottomLeft(6),  
1100 stapleTopRight(7),  
1101 stapleBottomRight(8),  
1102 saddleStitch(9),  
1103 edgeStitch(10),  
1104 punch(11),  
1105 cover(12),  
1106 bind(13)  
1107 }  
1108  
1109  
1110  
1111  
1112  
1113 **JmPrintQualityTC ::= TEXTUAL-CONVENTION**  
1114 STATUS current  
1115 DESCRIPTION  
1116 "Print quality settings."  
1117  
1118 These values are the same as the enum values of the IPP 'print-quality' attribute. See Section  
1119 3.6.1.2."

```

1120 REFERENCE
1121     "This is a type 2 enumeration. See Section 3.6.1.2."
1122 SYNTAX  INTEGER {
        other(1),          -- Not one of the specified or registered values.
                           --
        unknown(2),       -- The actual value is unknown.
        draft(3),         -- Lowest quality available on the printer.
        normal(4),        -- Normal or intermediate quality on the printer.
                           --
        high(5)           -- Highest quality available on the printer.
    }

```

```

1123 }
1124
1125
1126
1127
1128 JmPrinterResolutionTC ::= TEXTUAL-CONVENTION

```

```

1129     STATUS current
1130     DESCRIPTION

```

```

1131         "Printer resolutions.

```

```

1132
1133         Nine octets consisting of two 4-octet SIGNED-INTEGERS followed by a SIGNED-BYTE.
1134         The values are the same as those specified in the Printer MIB [printmib]. The first SIGNED-
1135         INTEGER contains the value of prtMarkerAddressabilityXFeedDir. The second SIGNED-
1136         INTEGER contains the value of prtMarkerAddressabilityFeedDir. The SIGNED-BYTE
1137         contains the value of prtMarkerAddressabilityUnit.

```

```

1138
1139         Note: the latter value is either 3 (tenThousandsOfInches) or 4 (micrometers) and the
1140         addressability is in 10,000 units of measure. Thus the SIGNED-INTEGERS represent integral
1141         values in either dots-per-inch or dots-per-centimeter. The values represent single integer
1142         resolutions or pairs of integer resolutions. The latter are to specify the resolution when the x
1143         and y dimensions differ. When two integers are specified, the first is in the x direction, i.e., in
1144         the direction of the shortest dimension of the medium, so that the value is independent of
1145         whether the printer feeds long edge or short edge first.

```

```

1146
1147         The syntax isse values are the same as the enum values of the IPP 'printer-resolution' attribute.
1148         See Section 3.6.1.2."

```

```

1149 REFERENCE

```

```

1150     "This is a type 2 enumeration. See Section ."

```

```

1151 SYNTAX  OCTET STRING (SIZE(9))INTEGER {
        other(1),          -- Not one of the specified or registered values.
                           --
        unknown(2),       -- The actual value is unknown.
        normal(3),        -- Normal resolution.
        res100(4),         -- 100 x 100 dpi
        res200(5),         -- 200 x 200 dpi
        res240(6),         -- 240 x 240 dpi
        res300(7),         -- 300 x 300 dpi
        res360(8),         -- 360 x 360 dpi
    }

```

```

res400(9);           -- 400 x 400 dpi
res600(10);        -- 600 x 600 dpi
res720(11);       -- 720 x 720 dpi
res800(12);       -- 800 x 800 dpi
res1200(13);      -- 1200 x 1200 dpi
res1440(14);     -- 1440 x 1440 dpi
res1600(15);    -- 1600 x 1600 dpi
res1800(16);    -- 1800 x 1800 dpi

-- future equal resolutions will be added here, the enum
-- values will not be re-sorted or re-assigned:
--
res100x200(100); -- 100 x 200 dpi
res200x100(101); -- 200 x 100 dpi
res300x600(102); -- 300 x 600 dpi
res600x300(103); -- 600 x 300 dpi
res360x720(104); -- 360 x 720 dpi
res720x360(105); -- 720 x 360 dpi
res400x800(106); -- 400 x 800 dpi
res800x400(107); -- 800 x 400 dpi
res600x1200(108); -- 600 x 1200 dpi
res1200x600(109); -- 1200 x 600 dpi
res720x1440(110); -- 720 x 1440 dpi
res1440x720(111); -- 1440 x 720 dpi
res1800x600(112); -- 1800 x 600 dpi

-- future unequal resolutions will be added here, the enum
-- values will not be re-sorted or re-assigned:
--

```

```

1152     †
1153
1154
1155
1156
1157
1158 JmTonerEconomyTC ::= TEXTUAL-CONVENTION
1159     STATUS current
1160     DESCRIPTION
1161         "Toner economy settings."
1162     REFERENCE
1163         "This is a type 2 enumeration. See Section 3.6.1.2."
1164     SYNTAX INTEGER {
1165         unknown(2),      -- unknown.
1166         off(3),         -- Off. Normal. Use full toner.
1167         on(4)          -- On. Use less toner than normal.

```

```

1168
1169
1170
1171 JmBooleanTC ::= TEXTUAL-CONVENTION
1172     STATUS      current
1173     DESCRIPTION
1174         "Boolean true or false value."
1175     REFERENCE
1176         "This is a type 2 enumeration.  See Section 3.6.1.2."
1177     SYNTAX      INTEGER {
1178         unknown(2),      -- unknown.
1179         false(3),       -- FALSE.
1180         true(4)         -- TRUE.
1181     }
1182
1183
1184 JmMediumTypeTC ::= TEXTUAL-CONVENTION
1185     STATUS      current
1186     DESCRIPTION
1187         "Identifies the type of medium."
1188
1189         other(1),
1190             The type is neither one of the values listed in this specification nor a registered value.
1191
1192         unknown(2),
1193             The type is not known.
1194
1195         stationery(3),
1196             Separately cut sheets of an opaque material.
1197
1198         transparency(4),
1199             Separately cut sheets of a transparent material.
1200
1201         envelope(5),
1202             Envelopes that can be used for conventional mailing purposes.
1203
1204         envelopePlain(6),
1205             Envelopes that are not preprinted and have no windows.
1206
1207         envelopeWindow(7),
1208             Envelopes that have windows for addressing purposes.
1209
1210         continuousLong(8),
1211             Continuously connected sheets of an opaque material connected along the long edge.

```

1212  
 1213       **continuousShort(9),**  
 1214           Continuously connected sheets of an opaque material connected along the short edge.  
 1215  
 1216       **tabStock(10),**  
 1217           Media with tabs.  
 1218  
 1219       **multiPartForm(11),**  
 1220           Form medium composed of multiple layers not pre-attached to one another; each sheet  
 1221           MAY be drawn separately from an input source.  
 1222  
 1223       **labels(12),**  
 1224           Label-stock.  
 1225  
 1226       **multiLayer(13)**  
 1227           Form medium composed of multiple layers which are pre-attached to one another, e.g. for  
 1228           use with impact printers."  
 1229 REFERENCE  
 1230       "This is a type 2 enumeration. See Section 3.6.1.2."  
 1231 SYNTAX   INTEGER {  
 1232       other(1),  
 1233       unknown(2),  
 1234       stationery(3),  
 1235       transparency(4),  
 1236       envelope(5),  
 1237       envelopePlain(6),  
 1238       envelopeWindow(7),  
 1239       continuousLong(8),  
 1240       continuousShort(9),  
 1241       tabStock(10),  
 1242       multiPartForm(11),  
 1243       labels(12),  
 1244       multiLayer(13)  
 1245       }  
 1246  
 1247  
 1248  
 1249  
 1250  
 1251 **JmJobSubmissionTypeTC ::= TEXTUAL-CONVENTION**  
 1252       STATUS   current  
 1253       DESCRIPTION  
 1254           "Identifies the format type of a job submission ID.  
 1255  
 1256           The ASCII characters '0-9', 'A-Z', and 'a-z' are assigned in order giving 62 possible formats.  
 1257  
 1258           Each job submission ID is a fixed-length, 48-octet printable ASCII coded character string,  
 1259           consisting of the following fields:

1260  
 1261 octet 1 The format letter.  
 1262 octets 2-40 A 39-character, ASCII trailing SPACE filled  
 1263 field specified by the format letter, if the  
 1264 data is less than 39 ASCII characters.  
 1265 octets 41-48 A sequential or random number to make the ID  
 1266 quasi-unique.  
 1267

1268 If the client does not supply a job submission ID in the job submission protocol, then the server  
 1269 SHALL assign a job submission ID using any of the standard formats that are reserved to the  
 1270 agent and adding the final 8 octets to distinguish the ID from others submitted from the same  
 1271 client. Clients SHALL not use formats that are reserved to agents.  
 1272

1273 The format values defined at the time of completion of the specification registered so far are:  
 1274

Format Letter	Description
'0'	octets 2-40: last 39 bytes of the <b>jmJobOwner</b> object. octets 41-48: 8-decimal-digit sequential number <u>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'.</u>

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

1309 as POSIX user number.  
 1310 octets 41-48: 8-decimal-digit sequential number  
 1311  
 1312 '6' octets 2-40: last 39 bytes of the user account  
 1313 number.  
 1314 octets 41-48: 8-decimal-digit sequential number  
 1315  
 1316 '7' octets 2-40: last 39 bytes of the DTMF incoming  
 1317 FAX routing number.  
 1318 octets 41-48: 8-decimal-digit sequential number  
 1319  
 1320 '8' octets 2-40: last 39 bytes of the job owner name  
 1321 (that the agent returns in the **jmJobOwner** object).  
 1322 octets 41-48: 8-decimal-digit sequential number  
 1323

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'

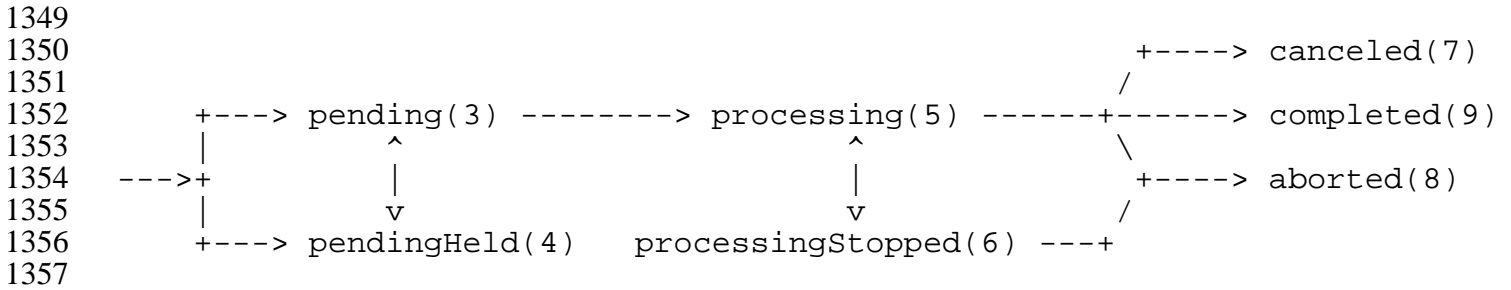
1343 **JmJobStateTC** ::= TEXTUAL-CONVENTION

1344 STATUS current

1345 DESCRIPTION

1346 "The current state of the job (**pending, processing, completed**, etc.).

1347  
 1348 The following figure shows the normal job state transitions:



1358 **Figure 4 - Normal Job State Transitions**

1359 Normally a job progresses from left to right. Other state transitions are unlikely, but are not  
1360 forbidden. Not shown are the transitions to the **canceled** state from the **pending**,  
1361 **pendingHeld**, **processing**, and **processingStopped** states.

1362 Jobs in the **pending**, **processing**, and **processingStopped** states are called 'active', while jobs in  
1363 the **pendingHeld**, **canceled**, **aborted**, and **completed** are called 'inactive'.

1364 These values are the same as the enum values of the IPP 'job-state' job attribute. See Section  
1365 3.6.1.2.

1366  
1367  
1368  
1369 **other(1),**  
1370 The job state is *not* one of the defined states.

1371  
1372  
1373 **unknown(2),**  
1374 The job state is *not* known, or its state is indeterminate.

1375  
1376 **pending(3),**  
1377 The job is a candidate to start processing, but is not yet processing.

1378  
1379 **pendingHeld(4),**  
1380 The job is not a candidate for processing for any number of reasons but will return to the  
1381 pending state as soon as the reasons are no longer present. The job's  
1382 **jmJobStateReasons1** object and/or **jobStateReasonsN** ( $N=2..4$ ) attributes SHALL  
1383 indicate why the job is no longer a candidate for processing. The reasons are represented  
1384 as bits in the **jmJobStateReasons1** object and/or **jobStateReasonsN** ( $N=2..4$ ) attributes.  
1385 See the **JmJobStateReasonsNTC** ( $N=1..4$ ) textual convention for the specification of  
1386 each reason.

1387  
1388 **processing(5),**  
1389 Either:  
1390  
1391 1. The job is using, or is attempting to use, one or more document transforms which  
1392 include (1) purely software processes that are interpreting a PDL, and (2) hardware  
1393 devices that are interpreting a PDL, making marks on a medium, and/or performing  
1394 finishing, such as stapling, etc.

1395 OR



1397  
1398  
1399 2. (configuration 2) the server has made the job ready for printing, but the output device is  
1400 not yet printing it, either because the job hasn't reached the output device or because the  
1401 job is queued in the output device or some other spooler, awaiting the output device to  
1402 print it.

1403 When the job is in the **processing** state, the entire job state includes the detailed status  
1404 represented in the device MIB indicated by the **hrDeviceIndex** value of the job's  
1405 **physicalDevice** attribute, if the agent implements such a device MIB.

1406  
1407 Implementations MAY, though they NEED NOT, include additional values in the job's  
1408 **jmJobStateReasons1** object to indicate the progress of the job, such as adding the  
1409 **jobPrinting** value to indicate when the device is actually making marks on a medium.

1410  
1411 **processingStopped(6),**  
1412 The job has stopped while processing for any number of reasons and will return to the  
1413 **processing** state as soon as the reasons are no longer present.

1414  
1415 The job's **jmJobStateReasons1** object and/or the job's **jobStateReasonsN** ( $N=2..4$ )  
1416 attributes MAY indicate why the job has stopped processing. For example, if the output  
1417 device is stopped, the **deviceStopped** value MAY be included in the job's  
1418 **jmJobStateReasons1** object.

1419  
1420 NOTE - When an output device is stopped, the device usually indicates its condition in  
1421 human readable form at the device. The management application can obtain more  
1422 complete device status remotely by querying the appropriate device MIB using the job's  
1423 **deviceIndex** attribute(s), if the agent implements such a device MIB

1424  
1425 **canceled(7),**  
1426 A client has canceled the job and the job is either: (1) in the process of being terminated by  
1427 the server or device or (2) has completed terminating. The job's **jmJobStateReasons1**  
1428 object SHOULD contain either the **canceledByUser** or **canceledByOperator** value.

1429  
1430 **aborted(8),**  
1431 The job has been aborted by the system, usually while the job was in the processing or  
1432 processingStopped state.

1433  
1434 **completed(9)**  
1435 The job has completed successfully or with warnings or errors after processing and all of  
1436 the media have been successfully stacked in the appropriate output bin(s). The job's  
1437 **jmJobStateReasons1** object SHOULD contain one of: **completedSuccessfully**,  
1438 **completedWithWarnings**, or **completedWithErrors** values."

1439 REFERENCE  
1440 "This is a type 2 enumeration. See Section 3.6.1.2."  
1441 SYNTAX INTEGER {  
1442 other(1),  
1443 unknown(2),  
1444 pending(3),  
1445 pendingHeld(4),

1446 processing(5),  
 1447 processingStopped(6),  
 1448 canceled(7),  
 1449 aborted(8),  
 1450 completed(9)  
 1451 }  
 1452

1453  
 1454 **JmAttributeTypeTC ::= TEXTUAL-CONVENTION**

1455 STATUS current  
 1456 DESCRIPTION

1457 "The type of the attribute which identifies the attribute.  
 1458

1459 In the following definitions of the enums, each description indicates whether the useful value of  
 1460 the attribute SHALL be represented using the **jmAttributeValueAsInteger** or the  
 1461 **jmAttributeValueAsOctets** objects by the initial tag: 'INTEGER:' or 'OCTETS:',  
 1462 respectively.  
 1463

1464 Some attributes allow the agent implementer a choice of useful values of either an integer, an  
 1465 octets representation, or both, depending on implementation. These attributes are indicated with  
 1466 'INTEGER:' AND/OR 'OCTETS:' tags.  
 1467

1468 A very few attributes require both objects at the same time to represent a pair of useful values  
 1469 (see **mediumConsumed(171)**). These attributes are indicated with 'INTEGER:' AND  
 1470 'OCTETS:' tags. See the **jmAttributeGroup** for the descriptions of these two MANDATORY  
 1471 objects.  
 1472

1473 NOTE - The enum assignments are grouped logically with values assigned in groups of 20, so  
 1474 that additional values may be registered in the future and assigned a value that is part of their  
 1475 logical grouping.  
 1476

1477 NOTE: No attribute name exceeds 31 characters.  
 1478

1479 ~~In the following descriptions of each attribute, the tags: 'INTEGER:' or 'OCTETS:' specify~~  
 1480 ~~whether the value SHALL be represented in the **jmAttributeValueAsInteger** or the~~  
 1481 ~~**jmAttributeValueAsOctets** object, or both, respectively.~~  
 1482

1483 The standard attribute types defined at the time of completion of the specification~~defined so far~~  
 1484 are:  
 1485

1486 **jmAttributeTypeIndex**

**Datatype**

1487 -----

-----

1488  
 1489 **other(1),**

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

**AND/OR**

**OCTET STRING(SIZE(0..63))**

1490  
 1491  
 1492 INTEGER: and/or OCTETS: An attribute that is not in the list and/or that has not been  
 1493 approved and registered with IANA.

1494  
1495  
1496  
1497  
1498  
1499  
1500  
1501  
1502  
1503  
1504  
1505  
1506  
1507  
1508  
1509  
1510  
1511  
1512  
1513  
1514  
1515  
1516  
1517  
1518  
1519  
1520  
1521  
1522  
1523  
1524  
1525  
1526  
1527  
1528  
1529  
1530  
1531  
1532  
1533  
1534  
1535  
1536  
1537  
1538  
1539  
1540  
1541  
1542

+++++

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

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

**processingMessage(6),** **OCTET STRING(SIZE(0..63))**  
 OCTETS: MULTI-ROW: A coded character set message that is generated during the processing of the job as a simple form of processing log to show progress and any problems.

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

+++++

+ **Job Identification attributes**

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

+++++

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

NOTE: This attribute NEED NOT be printable characters.

**serverAssignedJobName(22),** **OCTET STRING(SIZE(0..63))**  
 OCTETS: Configuration 3 only: The human readable string name, number, or ID of the

1543 job as assigned by the server that submitted the job to the device that the agent is  
 1544 providing access to with this MIB.

1545  
 1546 NOTE - This attribute is intended for enabling a user to find his/her job that a server  
 1547 submitted to a device when either the client does not support the **jmJobSubmissionID** or  
 1548 the server does not pass the **jmJobSubmissionID** through to the device.  
 1549

1550 **jobName(23),** **OCTET STRING(SIZE(0..63))**

1551 OCTETS: The human readable string name of the job as assigned by the submitting user  
 1552 to help the user distinguish between his/her various jobs. This name does not need to be  
 1553 unique.  
 1554

1555 This attribute is intended for enabling a user or the user's application to convey a job name  
 1556 that MAY be printed on a start sheet, returned in a **query** result, or used in notification or  
 1557 logging messages.  
 1558

1559 In order to assist users to find their jobs for job submission protocols that don't supply a  
 1560 **jmJobSubmissionID**, the agent SHOULD maintain the **jobName** attribute for the time  
 1561 specified by the **jmGeneralJobPersistence** object, rather than the (shorter)  
 1562 **jmGeneralAttributePersistence** object.  
 1563

1564 If this attribute is not specified when the job is submitted, no job name is assumed, but  
 1565 implementation specific defaults are allowed, such as the value of the **documentName**  
 1566 attribute of the first document in the job or the **fileName** attribute of the first document in  
 1567 the job.  
 1568

1569 The **jobName** attribute is distinguished from the **jobComment** attribute, in that the  
 1570 **jobName** attribute is intended to permit the submitting user to distinguish between  
 1571 different jobs that he/she has submitted. The **jobComment** attribute is intended to be free  
 1572 form additional information that a user might wish to use to communicate with  
 1573 himself/herself, such as a reminder of what to do with the results or to indicate a different  
 1574 set of input parameters were tried in several different job submissions.  
 1575

1576 **jobServiceTypes(24),** **JmJobServiceTypesTC**

1577 INTEGER: Specifies the type(s) of service to which the job has been submitted (print,  
 1578 fax, scan, etc.). The service type is bit encoded with each job service type so that more  
 1579 general and arbitrary services can be created, such as services with more than one  
 1580 destination type, or ones with only a source or only a destination. For example, a job  
 1581 service might **scan**, **faxOut**, and **print** a single job. In this case, three bits would be set in  
 1582 the **jobServiceTypes** attribute, corresponding to the hexadecimal values: **0x8** + **0x20** +  
 1583 **0x4**, respectively, yielding: **0x2C**.  
 1584

1585 Whether this attribute is set from a job attribute supplied by the job submission client or is  
 1586 set by the recipient job submission server or device depends on the job submission  
 1587 protocol. This attribute SHALL be implemented if the server or device has other types in  
 1588 addition to or instead of printing.  
 1589

1590 One of the purposes of this attribute is to permit a requester to filter out jobs that are not  
 1591 of interest. For example, a printer operator may only be interested in jobs that include  
 1592 printing.  
 1593

1594 **jobSourceChannelIndex(25),** Integer32(0..2147483647)  
 1595 INTEGER: The index of the row in the associated Printer MIB[print-mib] of the channel  
 1596 which is the source of the print job.  
 1597

1598 **jobSourcePlatformType(26),** **JmJobSourcePlatformTypeTC**  
 1599 INTEGER: The source platform type of the immediate upstream submitter that submitted  
 1600 the job to the server (configuration 2) or device (configuration 1 and 3) to which the agent  
 1601 is providing access. For configuration 1, this is the type of the client that submitted the  
 1602 job to the device; for configuration 2, this is the type of the client that submitted the job  
 1603 to the server; and for configuration 3, this is the type of the server that submitted the job  
 1604 to the device.  
 1605

1606 **submittingServerName(27),** **OCTET STRING(SIZE(0..63))**  
 1607 OCTETS: For configuration 3 only: The administrative name of the server that submitted  
 1608 the job to the device.  
 1609

1610 **submittingApplicationName(28),** **OCTET STRING(SIZE(0..63))**  
 1611 OCTETS: The name of the client application (not the server in configuration 3) that  
 1612 submitted the job to the server or device.  
 1613

1614 **jobOriginatingHost(29),** **OCTET STRING(SIZE(0..63))**  
 1615 OCTETS: The name of the client host (not the server host name in configuration 3) that  
 1616 submitted the job to the server or device.  
 1617

1618 **deviceNameRequested(30),** **OCTET STRING(SIZE(0..63))**  
 1619 OCTETS: The administratively defined coded character set name of the target device  
 1620 requested by the submitting user. For configuration 1, its value corresponds to the Printer  
 1621 MIB[print-mib]: **prtGeneralPrinterName** object. For configuration 2 and 3, its value is  
 1622 the name of the logical or physical device that the user supplied to indicate to the server  
 1623 on which device(s) they wanted the job to be processed.  
 1624

1625 **queueNameRequested(31),** **OCTET STRING(SIZE(0..63))**  
 1626 OCTETS: The administratively defined coded character set name of the target queue  
 1627 requested by the submitting user. For configuration 1, its value corresponds to the queue  
 1628 in the device for which the agent is providing access. For configuration 2 and 3, its value  
 1629 is the name of the queue that the user supplied to indicate to the server on which device(s)  
 1630 they wanted the job to be processed.  
 1631

1632 NOTE - typically an implementation SHOULD support either the **deviceNameRequested**  
 1633 or **queueNameRequested** attribute, but not both.  
 1634

1635 **physicalDevice(32),** **hrDeviceIndex**-(see HR-MIB)  
 1636 AND/OR  
 1637 **OCTET STRING(SIZE(0..63))**  
 1638 INTEGER: MULTI-ROW: The index of the physical device MIB instance

1639 requested/used, such as the Printer MIB[print-mib]. This value is an **hrDeviceIndex**  
 1640 value. See the Host Resources MIB[hr-mib].  
 1641  
 1642 AND/OR  
 1643  
 1644 OCTETS: MULTI-ROW: The name of the physical device to which the job is assigned.  
 1645  
 1646 **numberOfDocuments(33), Integer32(-2..2147483647)**  
 1647 INTEGER: The number of documents in this job.  
 1648  
 1649 **fileName(34), OCTET STRING(SIZE(0..63))**  
 1650 OCTETS: MULTI-ROW: The coded character set file name or URI[URI-spec] of the  
 1651 document.  
 1652  
 1653 There is no restriction on the same file name occurring in multiple rows.  
 1654  
 1655 **documentName(35), OCTET STRING(SIZE(0..63))**  
 1656 OCTETS: MULTI-ROW: The coded character set name of the document.  
 1657  
 1658 There is no restriction on the same document name occurring in multiple rows.  
 1659  
 1660 **jobComment(36), OCTET STRING(SIZE(0..63))**  
 1661 OCTETS: An arbitrary human-readable coded character text string supplied by the  
 1662 submitting user or the job submitting application program for any purpose. For example,  
 1663 a user might indicate what he/she is going to do with the printed output or the job  
 1664 submitting application program might indicate how the document was produced.  
 1665  
 1666 The **jobComment** attribute is not intended to be a name; see the **jobName** attribute.  
 1667  
 1668 **documentFormatIndex(37), Integer32(0..2147483647)**  
 1669 INTEGER: MULTI-ROW: The index in the **prtInterpreterTable** in the Printer  
 1670 MIB[print-mib] of the page description language (PDL) or control language interpreter  
 1671 that this job requires/uses. A document or a job MAY use more than one PDL or control  
 1672 language.  
 1673  
 1674 NOTE - As with all intensive attributes where multiple rows are allowed, there SHALL be  
 1675 only one distinct row for each distinct interpreter; there SHALL be no duplicates.  
 1676  
 1677 NOTE - This attribute type is intended to be used with an agent that implements the  
 1678 Printer MIB and SHALL not be used if the agent does not implement the Printer MIB.  
 1679 Such an agent SHALL use the **documentFormat** attribute instead.  
 1680  
 1681 **documentFormat(38), PrtInterpreterLangFamilyTC**  
 1682 AND/OR  
 1683 **OCTET STRING(SIZE(0..63))**  
 1684 INTEGER: MULTI-ROW: The interpreter language family corresponding to the Printer  
 1685 MIB[print-mib] **prtInterpreterLangFamily** object, that this job requires/uses. A  
 1686 document or a job MAY use more than one PDL or control language.  
 1687

1688  
1689  
1690  
1691  
1692  
1693  
1694  
1695  
1696  
1697  
1698  
1699  
1700  
1701  
1702  
1703  
1704  
1705  
1706  
1707  
1708  
1709  
1710  
1711  
1712  
1713  
1714  
1715  
1716  
1717  
1718  
1719  
1720  
1721  
1722  
1723  
1724  
1725  
1726  
1727  
1728  
1729  
1730  
1731  
1732  
1733  
1734  
1735

AND/OR

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

++++  
+ **Job Parameter attributes**  
+  
+ **The following attributes represent input parameters**  
+ **supplied by the submitting client in the job submission**  
+ **protocol.**  
++++

**jobPriority(50), Integer32(1..100)**

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

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

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

OCTETS: The calendar date and time of day after which the job SHALL become a candidate to be scheduled for processing. If the value of this attribute is in the future, the server SHALL set the value of the job's **jmJobState** object to **pendingHeld** and add the **jobProcessAfterSpecified** bit value to the job's **jmJobStateReasons1** object. When the specified date and time arrives, the server SHALL remove the **jobProcessAfterSpecified** bit value from the job's **jmJobStateReasons1** object and, if no other reasons remain, SHALL change the job's **jmJobState** object to **pending**.

**jobHold(52), JmBooleanTC**

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

**jobHoldUntil(53), OCTET STRING(SIZE(0..63))**

OCTETS: The named time period during which the job SHALL become a candidate for processing, such as ~~'no-hold'~~, **'evening'**, **'night'**, **'weekend'**, **'second-shift'**, **'third-shift'**, etc., as defined by the system administrator. See IPP [ipp-model] for the standard keyword values. Until that time period arrives, the job SHALL be in the **pendingHeld** state with the **jobHoldUntilSpecified** value in the **jmJobStateReasons1** object. The value **'no-hold'** SHALL indicate explicitly that no time period has been specified.

1736 **outputBin(54),** **Integer32(0..2147483647)**  
1737 **AND/OR**  
1738 **OCTET STRING(SIZE(0..63))**  
1739 INTEGER: MULTI-ROW: The output subunit index in the Printer MIB[print-mib]  
1740  
1741 **AND/OR**  
1742  
1743 OCTETS: the name or number (represented as ASCII digits) of the output bin to which  
1744 all or part of the job is placed in.  
1745  
1746 **sides(55),** **Integer32(-2..2)**  
1747 INTEGER: MULTI-ROW: The number of sides, '1' or '2', that any document in this job  
1748 requires/used.  
1749  
1750 **finishing(56),** **JmFinishingTC**  
1751 INTEGER: MULTI-ROW: Type of finishing that any document in this job requires/used.  
1752  
1753  
1754 ++++++  
1755 + **Image Quality attributes (requested and consumed)**  
1756 +  
1757 + **For devices that can vary the image quality.**  
1758 ++++++  
1759  
1760 **printQualityRequested(70),** **JmPrintQualityTC**  
1761 INTEGER: MULTI-ROW: The print quality selection requested for a document in the  
1762 job for printers that allow quality differentiation.  
1763  
1764 **printQualityUsed(71),** **JmPrintQualityTC**  
1765 INTEGER: MULTI-ROW: The print quality selection actually used by a document in the  
1766 job for printers that allow quality differentiation.  
1767  
1768 **printerResolutionRequested(72),** **JmPrinterResolutionTC**  
1769 OCTETSINTEGER: MULTI-ROW: The printer resolution requested for a document in  
1770 the job for printers that support resolution selection.  
1771  
1772 **printerResolutionUsed(73),** **JmPrinterResolutionTC**  
1773 OCTETSINTEGER: MULTI-ROW: The printer resolution actually used by a document  
1774 in the job for printers that support resolution selection.  
1775  
1776 **tonerEcomonyRequested(74),** **JmTonerEcomonyTC**  
1777 INTEGER: MULTI-ROW: The print quality selection requested for documents in the  
1778 job for printers that allow toner quality differentiation.  
1779  
1780 **tonerEcomonyUsed(75),** **JmTonerEcomonyTC**  
1781 INTEGER: MULTI-ROW: The print quality selection actually used by documents in the  
1782 job for printers that allow toner quality differentiation.  
1783



1784  
1785  
1786  
1787  
1788  
1789  
1790  
1791  
1792  
1793  
1794  
1795  
1796  
1797  
1798  
1799  
1800  
1801  
1802  
1803  
1804  
1805  
1806  
1807  
1808  
1809  
1810  
1811  
1812  
1813  
1814  
1815  
1816  
1817  
1818  
1819  
1820  
1821  
1822  
1823  
1824  
1825  
1826  
1827  
1828  
1829  
1830  
1831  
1832

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

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

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

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

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

**documentCopiesRequested(92), Integer32(-2..2147483647)**  
 INTEGER: The total count of the number of document copies requested. If there are documents A, B, and C, and document B is specified to produce 4 copies, the number of document copies requested is 6 for the job.

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

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

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

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

1833  
1834  
1835  
1836  
1837  
1838  
1839  
1840  
1841  
1842  
1843  
1844  
1845  
1846  
1847  
1848  
1849  
1850  
1851  
1852  
1853  
1854  
1855  
1856  
1857  
1858  
1859  
1860  
1861  
1862  
1863  
1864  
1865  
1866  
1867  
1868  
1869  
1870  
1871  
1872  
1873  
1874  
1875  
1876  
1877  
1878  
1879  
1880

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

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

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

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

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

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

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

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

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

1881  
1882  
1883  
1884  
1885  
1886  
1887  
1888  
1889  
1890  
1891  
1892  
1893  
1894  
1895  
1896  
1897  
1898  
1899  
1900  
1901  
1902  
1903  
1904  
1905  
1906  
1907  
1908  
1909  
1910  
1911  
1912  
1913  
1914  
1915  
1916  
1917  
1918  
1919  
1920  
1921  
1922  
1923  
1924  
1925  
1926  
1927  
1928  
1929

**highlightColorImpressionsCompleted(115), Integer32(-2..2147483647)**

INTEGER: The number of highlight color impressions completed by the device for this job so far. For printing, the impressions completed includes interpreting, marking, and stacking the output. For other types of job services, the number of impressions completed includes the number of impressions processed. Highlight color impressions are typically defined as those requiring black plus one other colorant, but this MAY vary by implementation.

++++  
+ **Page attributes**

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

++++  
**pagesRequested(130), Integer32(-2..2147483647)**

INTEGER: The number of logical pages requested by the job to be processed.

**pagesCompleted(131), Integer32(-2..2147483647)**

INTEGER: The number of logical pages completed for this job so far.

**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.

1930  
1931  
1932  
1933  
1934  
1935  
1936  
1937  
1938  
1939  
1940  
1941  
1942  
1943  
1944  
1945  
1946  
1947  
1948  
1949  
1950  
1951  
1952  
1953  
1954  
1955  
1956  
1957  
1958  
1959  
1960  
1961  
1962  
1963  
1964  
1965  
1966  
1967  
1968  
1969  
1970  
1971  
1972  
1973  
1974  
1975  
1976  
1977  
1978

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  
OCTET STRING(SIZE(0..63))  
INTEGER: MULTI-ROW: The type  
AND/OR  
OCTETS: the name of the medium that is required by the job.

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

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

**colorantConsumed(173),** **Integer32(-2..2147483647)**  
AND/OR  
OCTET STRING(SIZE(0..63))  
INTEGER: MULTI-ROW: The index (**prtMarkerColorantIndex**) in the Printer  
MIB[print-mib]  
AND/OR  
OCTETS: the name of the colorant consumed.

++++  
+ Time attributes (set by server or device)

1979 +  
 1980 + This section of attributes are ones that are set by the  
 1981 + server or device that accepts jobs. Two forms of time are  
 1982 + provided. Each form is represented in a separate attribute.  
 1983 + See section 3.1.2 and section 3.1.3 for the  
 1984 + conformance requirements for time attribute for agents and  
 1985 + monitoring applications, respectively. The two forms are:  
 1986 +  
 1987 + 'DateAndTime' is an 8 or 11 octet binary encoded year,  
 1988 + month, day, hour, minute, second, deci-second with  
 1989 + optional offset from UTC. See SNMPv2-TC [SMIV2-TC].  
 1990 +  
 1991 + NOTE: 'DateAndTime' is not printable characters; it is  
 1992 + binary.  
 1993 +  
 1994 + 'JmTimeStampTC' is the time of day measured in the number of  
 1995 + seconds since the system was booted.  
 1996 ++++++  
 1997  
 1998 **jobSubmissionToServerTime(190),** **JmTimeStampTC**  
 1999 AND/OR  
 2000 **DateAndTime**-(SNMPv2-TC)  
 2001 INTEGER: Configuration 3 only: The time  
 2002 AND/OR  
 2003 OCTETS: the date and time that the job was submitted to the server (as distinguished  
 2004 from the device which uses jobSubmissionTime).  
 2005  
 2006 **jobSubmissionTime(191),** **JmTimeStampTC**  
 2007 AND/OR  
 2008 **DateAndTime**-(SNMPv2-TC)  
 2009 INTEGER: Configurations 1, 2, and 3: The time  
 2010 AND/OR  
 2011 OCTETS: the date and time that the job was submitted to the server or device to which  
 2012 the agent is providing access.  
 2013  
 2014  
 2015  
 2016 **jobStartedBeingHeldTime(192),** **JmTimeStampTC**  
 2017 AND/OR  
 2018 **DateAndTime**-(SNMPv2-TC)  
 2019 INTEGER: The time  
 2020 AND/OR  
 2021 OCTETS: the date and time that the job last entered the **pendingHeld** state. If the job  
 2022 has never entered the **pendingHeld** state, then the value SHALL be '0' or the attribute  
 2023 SHALL not be present in the table.  
 2024  
 2025 **jobStartedProcessingTime(193),** **JmTimeStampTC**  
 2026 AND/OR  
 2027 **DateAndTime**-(SNMPv2-TC)

2028 INTEGER: The time  
 2029 AND/OR  
 2030 OCTETS: the date and time that the job started processing.  
 2031  
 2032 **jobCompletedTime(194),** **JmTimeStampTC**  
 2033 AND/OR  
 2034 **DateAndTime** (~~SNMPv2-TC~~)  
 2035 INTEGER: The time  
 2036 AND/OR  
 2037 OCTETS: the date and time that the job entered the **completed, canceled, or aborted**  
 2038 state.  
 2039  
 2040 **jobProcessingCPUtime(195)** **Integer32(-2..2147483647)**  
 2041 **UNITS 'seconds'**  
 2042 INTEGER: The amount of CPU time in seconds that the job has been in the **processing**  
 2043 state. If the job enters the **processingStopped** state, that elapsed time SHALL not be  
 2044 included. In other words, the **jobProcessingCPUtime** value SHOULD be relatively  
 2045 repeatable when the same job is processed again on the same device."  
 2046  
 2047 REFERENCE  
 2048 "See Section 3.2 entitled 'The Attribute Mechanism' for a description of this textual-convention  
 2049 and its use in the **jmAttributeTable**.  
 2050  
 2051 This is a type 2 enumeration. See Section 3.6.1.2."  
 2052 SYNTAX INTEGER {  
 2053 other(1),  
 2054 unknown(2),  
 2055 jobStateReasons2(3),  
 2056 jobStateReasons3(4),  
 2057 jobStateReasons4(5),  
 2058 processingMessage(6),  
 2059  
 2060 jobAccountName(21),  
 2061 serverAssignedJobName(22),  
 2062 jobName(23),  
 2063 jobServiceTypes(24),  
 2064 jobSourceChannelIndex(25),  
 2065 jobSourcePlatformType(26),  
 2066 submittingServerName(27),  
 2067 submittingApplicationName(28),  
 2068 jobOriginatingHost(29),  
 2069 deviceNameRequested(30),  
 2070 queueNameRequested(31),  
 2071 physicalDevice(32),  
 2072 numberOfDocuments(33),  
 2073 fileName(34),  
 2074 documentName(35),  
 2075 jobComment(36),  
 2076 documentFormatIndex(37),

2077 documentFormat(38),  
2078  
2079 jobPriority(50),  
2080 jobProcessAfterDateAndTime(51),  
2081 jobHold(52),  
2082 jobHoldUntil(53),  
2083 outputBin(54),  
2084 sides(55),  
2085 finishing(56),  
2086  
2087 printQualityRequested(70),  
2088 printQualityUsed(71),  
2089 printerResolutionRequested(72),  
2090 printerResolutionUsed(73),  
2091 tonerEcomonyRequested(74),  
2092 tonerEcomonyUsed(75),  
2093 tonerDensityRequested(76),  
2094 tonerDensityUsed(77),  
2095  
2096 jobCopiesRequested(90),  
2097 jobCopiesCompleted(91),  
2098 documentCopiesRequested(92),  
2099 documentCopiesCompleted(93),  
2100 jobKOctetsTransferred(94),  
2101  
2102 impressionsSpooled(110),  
2103 impressionsSentToDevice(111),  
2104 impressionsInterpreted(112),  
2105 impressionsCompletedCurrentCopy(113),  
2106 fullColorImpressionsCompleted(114),  
2107 highlightColorImpressionsCompleted(115),  
2108  
2109 pagesRequested(130),  
2110 pagesCompleted(131),  
2111 pagesCompletedCurrentCopy(132),  
2112  
2113 sheetsRequested(150),  
2114 sheetsCompleted(151),  
2115 sheetsCompletedCurrentCopy(152),  
2116  
2117 mediumRequested(170),  
2118 mediumConsumed(171),  
2119 colorantRequested(172),  
2120 colorantConsumed(173),  
2121  
2122 jobSubmissionToServerTime(190),  
2123 jobSubmissionTime(191),  
2124 jobStartedBeingHeldTime(192),  
2125 jobStartedProcessingTime(193),

```

2126     jobCompletedTime(194),
2127     jobProcessingCPUTime(195)
2128 }
2129
2130
2131

```

2132

**JmJobServiceTypesTC ::= TEXTUAL-CONVENTION**

2134 STATUS current

2135 DESCRIPTION

2136 "Specifies the type(s) of service to which the job has been submitted (print, fax, scan, etc.). The  
 2137 service type is represented as an enum that is bit encoded with each job service type so that  
 2138 more general and arbitrary services can be created, such as services with more than one  
 2139 destination type, or ones with only a source or only a destination. For example, a job service  
 2140 might **scan**, **faxOut**, and **print** a single job. In this case, three bits would be set in the  
 2141 **jobServiceTypes** attribute, corresponding to the hexadecimal values: **0x8** + **0x20** + **0x4**,  
 2142 respectively, yielding: **0x2C**.

2143

2144 Whether this attribute is set from a job attribute supplied by the job submission client or is set by  
 2145 the recipient job submission server or device depends on the job submission protocol. With  
 2146 either implementation, the agent SHALL return a non-zero value for this attribute indicating the  
 2147 type of the job.

2148

2149 One of the purposes of this attribute is to permit a requester to filter out jobs that are not of  
 2150 interest. For example, a printer operator MAY only be interested in jobs that include printing.  
 2151 That is why the attribute is in the job identification category.

2152

2153 The following service component types are defined (in hexadecimal) and are assigned a separate  
 2154 bit value for use with the **jobServiceTypes** attribute:

2155

**other 0x1**

2157 The job contains some instructions that are not one of the identified types.

2158

**unknown****0x2**

2160 The job contains some instructions whose type is unknown to the agent.

2161

**print 0x4**

2163 The job contains some instructions that specify printing

2164

**scan 0x8**

2166 The job contains some instructions that specify scanning

2167

**faxIn 0x10**

2169 The job contains some instructions that specify receive fax

2170

**faxOut****0x20**

2172 The job contains some instructions that specify sending fax

2173



2174           **getFile**                                       **0x40**  
 2175           The job contains some instructions that specify accessing files or documents  
 2176

2177           **putFile**                                       **0x80**  
 2178           The job contains some instructions that specify storing files or documents  
 2179

2180           **mailList**                                       **0x100**  
 2181           The job contains some instructions that specify distribution of documents using an  
 2182           electronic mail system."  
 2183 REFERENCE  
 2184           "These bit definitions are the equivalent of a type 2 enum except that combinations of them  
 2185           MAY be used together. See section 3.6.1.2."  
 2186 SYNTAX   **INTEGER(0..2147483647)** -- 31 bits, all but sign bit  
 2187

2188

2189

2190

2191 **JmJobStateReasons1TC ::= TEXTUAL-CONVENTION**  
 2192       STATUS     current  
 2193       DESCRIPTION  
 2194       "The **JmJobStateReasonsN**TC ( $N=1..4$ ) textual-conventions are used with the  
 2195       **jmJobStateReasons1** object and **jobStateReasonsN** ( $N=2..4$ ), respectively, to provides  
 2196       additional information regarding the current **jmJobState** object value. These values MAY be  
 2197       used with any job state or states for which the reason makes sense.  
 2198

2199       NOTE - While values cannot be added to the **jmJobState** object without impacting deployed  
 2200       clients that take actions upon receiving **jmJobState** values, it is the intent that additional  
 2201       **JmJobStateReasonsN**TC enums can be defined and registered without impacting such  
 2202       deployed clients. In other words, the **jmJobStateReasons1** object and **jobStateReasonsN**  
 2203       attributes are intended to be extensible.  
 2204

2205       NOTE - The Job Monitoring MIB contains a superset of the IPP values[ipp-model] for the IPP  
 2206       'job-state-reasons' attribute, since the Job Monitoring MIB is intended to cover other job  
 2207       submission protocols as well. Also some of the names of the reasons have been changed from  
 2208       'printer' to 'device', since the Job Monitoring MIB is intended to cover additional types of  
 2209       devices, including input devices, such as scanners.  
 2210

2211       The following standard values are defined (in hexadecimal) as *powers of two*, since multiple  
 2212       values MAY be used at the same time. For ease of understanding, the  
 2213       **JmJobStateReasons1TC** reasons are presented in the order in which the reasons ~~are~~<sup>are most</sup>  
 2214       likely to occur (if implemented), starting with the '**jobIncoming**' value and ending with  
 2215       '**jobCompletedWithErrors**' reasons ~~not counting the 'other' and 'unknown' reasons.~~  
 2216

2217           **other**                                       **0x1**  
 2218           The job state reason is not one of the standardized or registered reasons.  
 2219

2220	<b>unknown</b>	<b>0x2</b>
2221	The job state reason is not known to the agent or is indeterminent.	
2222		
2223	<b>jobIncoming</b>	<b>0x4</b>
2224	The job has been accepted by the server or device, but the server or device is expecting	
2225	(1) additional operations from the client to finish creating the job and/or (2) is	
2226	accessing/accepting document data.	
2227		
2228	<b>jobOutgoing</b>	<b>0x8</b>
2229	Configuration 2 only: The server is transmitting the job to the device.	
2230		
2231	<b>jobHoldSpecified</b>	<b>0x10</b>
2232	The value of the job's <b>jobHold(52)</b> attribute is TRUE. The job SHALL NOT be a	
2233	candidate for processing until this reason is removed and there are no other reasons to	
2234	hold the job.	
2235		
2236	<b>jobHoldUntilSpecified</b>	<b>0x20</b>
2237	The value of the job's <b>jobHoldUntil(53)</b> attribute specifies a time period that is still in the	
2238	future. The job SHALL NOT be a candidate for processing until this reason is removed	
2239	and there are no other reasons to hold the job.	
2240		
2241	<b>jobProcessAfterSpecified</b>	<b>0x40</b>
2242	The value of the job's <b>jobProcessAfterDateAndTime(51)</b> attribute specifies a time that is	
2243	still in the future, either set when the job was created or subsequently by an explicit modify	
2244	job operation. The job SHALL NOT be a candidate for processing until this reason is	
2245	removed and there are no other reasons to hold the job.	
2246		
2247	<b>resourcesAreNotReady</b>	<b>0x80</b>
2248	At least one of the resources needed by the job, such as media, fonts, resource objects,	
2249	etc., is not ready on any of the physical devices for which the job is a candidate. This	
2250	condition MAY be detected when the job is accepted, or subsequently while the job is	
2251	<b>pending</b> or <b>processing</b> , depending on implementation.	
2252		
2253	<b>deviceStoppedPartly</b>	<b>0x100</b>
2254	One or more, but not all, of the devices to which the job is assigned are stopped. If all of	
2255	the devices are stopped (or the only device is stopped), the <b>deviceStopped</b> reason	
2256	SHALL be used.	
2257		
2258	<b>deviceStopped</b>	<b>0x200</b>
2259	The device(s) to which the job is assigned is (are all) stopped.	
2260		
2261	<b>jobPrinting</b>	<b>0x400</b>
2262	The output device is marking media. This attribute is useful for servers and output devices	
2263	which spend a great deal of time processing when no marking is happening and then want	
2264	to show that marking is now happening or when the job is in the <b>canceled</b> or <b>aborted</b>	
2265	state, but the marking has not yet stopped so that impression or sheet counts are still	
2266	increasing for the job.	
2267		

2268	<b>jobCanceledByUser</b>	<b>0x800</b>
2269	The job was canceled by the user, i.e., by an unknown user or by a user whose name is the	
2270	same as the value of the job's <b>jmJobOwner</b> object.	
2271		
2272	<b>jobCanceledByOperator</b>	<b>0x1000</b>
2273	The job was canceled by the operator, i.e., by a user whose name is different than the	
2274	value of the job's <b>jmJobOwner</b> object.	
2275		
2276	<b>abortedBySystem</b>	<b>0x2000</b>
2277	The job was aborted by the system.	
2278		
2279	NOTE - <u>When the system puts a job into the 'aborted' job state, this reason is not needed.</u>	
2280	† <u>This reason is needed only when the system aborts a job, but, instead of placing does not</u>	
2281	<u>put the job in the <b>aborted</b> job state. For example, if the system aborts the job, but places</u>	
2282	<u>the job in the <b>pendingHeld</b> state, so that a user or operator can <u>manually</u> try the job</u>	
2283	<u>again.</u>	
2284		
2285	<b>jobCompletedSuccessfully</b>	<b>0x4000</b>
2286	The job completed successfully.	
2287		
2288	<b>jobCompletedWithWarnings</b>	<b>0x8000</b>
2289	The job completed with warnings.	
2290		
2291	<b>jobCompletedWithErrors</b>	<b>0x10000</b>
2292	The job completed with errors (and possibly warnings too).	
2293		
2294	The following additional job state reasons have been added to represent job states that are	
2295	in ISO DPA[iso-dpa] and other job submission protocols:	
2296		
2297	<b>jobPaused</b>	<b>0x20000</b>
2298	The job has been indefinitely suspended by a client issuing an operation to suspend the job	
2299	so that other jobs may proceed using the same devices. The client MAY issue an	
2300	operation to resume the paused job at any time, in which case the agent SHALL remove	
2301	the <b>jobPaused</b> values from the job's <b>jmJobStateReasons1</b> object and the job is eventually	
2302	resumed at or near the point where the job was paused.	
2303		
2304	<b>jobInterrupted</b>	<b>0x40000</b>
2305	The job has been interrupted while processing by a client issuing an operation that	
2306	specifies another job to be run instead of the current job. The server or device will	
2307	automatically resume the interrupted job when the interrupting job completes.	
2308		
2309	<b>jobRetained</b>	<b>0x80000</b>
2310	The job is being retained by the server or device with all of the job's document data (and	
2311	submitted resources, such as fonts, logos, and forms, if any). Thus a client could issue an	
2312	operation to the server or device to either (1) re-do the job (or a copy of the job) on the	
2313	same server or device or (2) resubmit the job to another server or device. When a client	
2314	could no longer re-do/resubmit the job, such as after the document data has been	
2315	discarded, the agent SHALL remove the <b>jobRetained</b> value from the	
2316	<b>jmJobStateReasons1</b> object."	

2317 REFERENCE  
 2318 "These bit definitions are the equivalent of a type 2 enum except that combinations of bits may  
 2319 be used together. See section 3.6.1.2. The remaining bits are reserved for future  
 2320 standardization and/or registration."  
 2321  
 2322 SYNTAX **INTEGER(0..2147483647)** -- 31 bits, all but sign bit  
 2323  
 2324  
 2325  
 2326  
 2327  
 2328 **JmJobStateReasons2TC** ::= TEXTUAL-CONVENTION  
 2329 STATUS current  
 2330 DESCRIPTION  
 2331 "This textual-convention is used with the **jobStateReasons2** attribute to provides additional  
 2332 information regarding the **jmJobState** object. See the description under  
 2333 **JmJobStateReasons1TC** for additional information that applies to all reasons.  
 2334  
 2335 The following standard values are defined (in hexadecimal) as *powers of two*, since multiple  
 2336 values may be used at the same time:  
 2337  
 2338 **cascaded** **0x1**  
 2339 An outbound gateway has transmitted all of the job's job and document attributes and data  
 2340 to another spooling system.  
 2341  
 2342 **deletedByAdministrator** **0x2**  
 2343 The administrator has deleted the job.  
 2344  
 2345 **discardTimeArrived** **0x4**  
 2346 The job has been deleted due to the fact that the time specified by the job's job-discard-  
 2347 time attribute has arrived.  
 2348  
 2349 **postProcessingFailed** **0x8**  
 2350 The post-processing agent failed while trying to log accounting attributes for the job;  
 2351 therefore the job has been placed into the completed state with the **jobRetained**  
 2352 **jmJobStateReasons1** object value for a system-defined period of time, so the  
 2353 administrator can examine it, resubmit it, etc.  
 2354  
 2355 **submissionInterrupted** **0x10**  
 2356 Indicates that the job was not completely submitted for some unforeseen reason, such as:  
 2357 (1) the server has crashed before the job was closed by the client, (2) the server or the  
 2358 document transfer method has crashed in some non-recoverable way before the document  
 2359 data was entirely transferred to the server, (3) the client crashed or failed to close the job  
 2360 before the time-out period.  
 2361  
 2362 **maxJobFaultCountExceeded** **0x20**  
 2363 The job has faulted several times and has exceeded the administratively defined fault count  
 2364 limit.  
 2365

2366	<b>devicesNeedAttentionTimeOut</b>	<b>0x40</b>
2367	One or more document transforms that the job is using needs human intervention in order	
2368	for the job to make progress, but the human intervention did not occur within the site-	
2369	settable time-out value.	
2370		
2371	<b>needsKeyOperatorTimeOut</b>	<b>0x80</b>
2372	One or more devices or document transforms that the job is using need a specially trained	
2373	operator (who may need a key to unlock the device and gain access) in order for the job to	
2374	make progress, but the key operator intervention did not occur within the site-settable	
2375	time-out value.	
2376		
2377	<b>jobStartWaitTimeOut</b>	<b>0x100</b>
2378	The server/device has stopped the job at the beginning of processing to await human	
2379	action, such as installing a special cartridge or special non-standard media, but the job was	
2380	not resumed within the site-settable time-out value and the server/device has transitioned	
2381	the job to the <b>pendingHeld</b> state.	
2382		
2383	<b>jobEndWaitTimeOut</b>	<b>0x200</b>
2384	The server/device has stopped the job at the end of processing to await human action,	
2385	such as removing a special cartridge or restoring standard media, but the job was not	
2386	resumed within the site-settable time-out value and the server/device has transitioned the	
2387	job to the completed state.	
2388		
2389	<b>jobPasswordWaitTimeOut</b>	<b>0x400</b>
2390	The server/device has stopped the job at the beginning of processing to await input of the	
2391	job's password, but the password was not received within the site-settable time-out value.	
2392		
2393	<b>deviceTimedOut</b>	<b>0x800</b>
2394	A device that the job was using has not responded in a period specified by the device's	
2395	site-settable attribute.	
2396		
2397	<b>connectingToDeviceTimeOut</b>	<b>0x1000</b>
2398	The server is attempting to connect to one or more devices which may be dial-up, polled,	
2399	or queued, and so may be busy with traffic from other systems, but server was unable to	
2400	connect to the device within the site-settable time-out value.	
2401		
2402	<b>transferring</b>	<b>0x2000</b>
2403	The job is being transferred to a down stream server or device.	
2404		
2405	<b>queuedInDevice</b>	<b>0x4000</b>
2406	The job has been queued in a down stream server or device.	
2407		
2408	<b>jobCleanup</b>	<b>0x8000</b>
2409	The server/device is performing cleanup activity as part of ending normal processing.	
2410		
2411	<b>processingToStopPoint</b>	<b>0x10000</b>
2412	The requester has issued an operation to interrupt the job and the server/device is	
2413	processing up until the specified stop point occurs.	
2414		

2415	<b>jobPasswordWait</b>	<b>0x20000</b>
2416	The server/device has selected the job to be next to process, but instead of assigning	
2417	resources and starting the job processing, the server/device has transitioned the job to the	
2418	<b>pendingHeld</b> state to await entry of a password (and dispatched another job, if there is	
2419	one).	
2420		
2421	<b>validating</b>	<b>0x40000</b>
2422	The server/device is validating the job <i>after</i> accepting the job.	
2423		
2424	<b>queueHeld</b>	<b>0x80000</b>
2425	The operator has held the entire job set or queue.	
2426		
2427	<b>jobProofWait</b>	<b>0x100000</b>
2428	The job has produced a single proof copy and is in the <b>pendingHeld</b> state waiting for the	
2429	requester to issue an operation to release the job to print normally, obeying any job and	
2430	document copy attributes that were originally submitted.	
2431		
2432	<b>heldForDiagnostics</b>	<b>0x200000</b>
2433	The system is running intrusive diagnostics, so that all jobs are being held.	
2434		
2435	<b>serviceOffLine</b>	<b>0x400000</b>
2436	The service/document transform is off-line and accepting no jobs. All pending jobs are put	
2437	into the pendingHeld state. This could be true if its input is impaired or broken.	
2438		
2439	<b>noSpaceOnServer</b>	<b>0x800000</b>
2440	There is no room on the server to store all of the job.	
2441		
2442	<b>pinRequired</b>	<b>0x1000000</b>
2443	The System Administrator settable device policy is (1) to require PINs, and (2) to hold	
2444	jobs that do not have a pin supplied as an input parameter when the job was created.	
2445		
2446	<b>exceededAccountLimit</b>	<b>0x2000000</b>
2447	The account for which this job is drawn has exceeded its limit. This condition SHOULD	
2448	be detected before the job is scheduled so that the user does not wait until his/her job is	
2449	scheduled only to find that the account is overdrawn. This condition MAY also occur	
2450	while the job is processing either as processing begins or part way through processing.	
2451		
2452	<b>heldForRetry</b>	<b>0x4000000</b>
2453	The job encountered some errors that the server/device could not recover from with its	
2454	normal retry procedures, but the error might not be encountered if the job is <u>processed</u>	
2455	<u>again</u> - <del>tried</del> in the future. <u>Example cases are</u> such as phone number busy or remote file	
2456	system in-accessible. For such a situation, the server/device SHALL transition the job	
2457	from the <b>processing</b> to the <b>pendingHeld</b> , rather than to the <b>aborted</b> state.	
2458		
2459	The following values are from the X/Open PSIS draft standard:	
2460		
2461	<b>canceledByShutdown</b>	<b>0x8000000</b>
2462	The job was canceled because the server or device was shutdown before completing the	
2463	job.	

2464  
 2465       **deviceUnavailable**                               **0x10000000**  
 2466               This job was aborted by the system because the device is currently unable to accept jobs.  
 2467

2468       **wrongDevice**                                       **0x20000000**  
 2469               This job was aborted by the system because the device is unable to handle this particular  
 2470               job; the spooler SHOULD try another device or the user should submit the job to another  
 2471               device.  
 2472

2473       **badJob**   **0x40000000**  
 2474               This job was aborted by the system because this job has a major problem, such as an ill-  
 2475               formed PDL; the spooler SHOULD not even try another device. "  
 2476 REFERENCE  
 2477               "These bit definitions are the equivalent of a type 2 enum except that combinations of them may  
 2478               be used together. See section 3.6.1.2. See the description under **JmJobStateReasons1TC** and  
 2479               the **jobStateReasons2** attribute."  
 2480

2481 SYNTAX    **INTEGER(0..2147483647)** -- 31 bits, all but sign bit  
 2482  
 2483  
 2484  
 2485  
 2486  
 2487

2488 **JmJobStateReasons3TC** ::= TEXTUAL-CONVENTION  
 2489       STATUS    current  
 2490       DESCRIPTION  
 2491               "This textual-convention is used with the **jobStateReasons3** attribute to provides additional  
 2492               information regarding the **jmJobState** object. See the description under  
 2493               **JmJobStateReasons1TC** for additional information that applies to all reasons.  
 2494  
 2495               The following standard values are defined (in hexadecimal) as *powers of two*, since multiple  
 2496               values may be used at the same time:  
 2497

2498       **jobInterruptedByDeviceFailure**                       **0x1**  
 2499               A device or the print system software that the job was using has failed while the job was  
 2500               processing. The server or device is keeping the job in the **pendingHeld** state until an  
 2501               operator can determine what to do with the job."  
 2502 REFERENCE  
 2503               "These bit definitions are the equivalent of a type 2 enum except that combinations of them may  
 2504               be used together. See section 3.6.1.2. The remaining bits are reserved for future  
 2505               standardization and/or registration. See the description under **JmJobStateReasons1TC** and the  
 2506               **jobStateReasons3** attribute."  
 2507 SYNTAX    **INTEGER(0..2147483647)** -- 31 bits, all but sign bit  
 2508  
 2509  
 2510  
 2511  
 2512

2513 **JmJobStateReasons4TC** ::= TEXTUAL-CONVENTION  
2514     STATUS     current  
2515     DESCRIPTION  
2516         "This textual-convention is used in the **jobStateReasons4** attribute to provides additional  
2517         information regarding the **jmJobState** object. See the description under  
2518         **JmJobStateReasons1TC** for additional information that applies to all reasons.  
2519  
2520         The following standard values are defined (in hexadecimal) as *powers of two*, since multiple  
2521         values may be used at the same time:  
2522  
2523         none yet defined. These bits are reserved for future standardization and/or registration."  
2524     REFERENCE  
2525         "These bit definitions are the equivalent of a type 2 enum except that combinations of them may  
2526         be used together. See section 3.6.1.2. See the description under **JmJobStateReasons1TC** and  
2527         the **jobStateReasons4** attribute."  
2528  
2529     SYNTAX     **INTEGER(0..2147483647)** -- 31 bits, all but sign bit



```

2530
2531 jobmonMIBObjects OBJECT IDENTIFIER ::= { jobmonMIB 1 }
2532
2533 -- The General Group (MANDATORY)
2534
2535 -- The jmGeneralGroup consists entirely of the jmGeneralTable.
2536
2537 jmGeneral OBJECT IDENTIFIER ::= { jobmonMIBObjects 1 }
2538
2539 jmGeneralTable OBJECT-TYPE
2540     SYNTAX      SEQUENCE OF JmGeneralEntry
2541     MAX-ACCESS  not-accessible
2542     STATUS      current
2543     DESCRIPTION
2544         "The jmGeneralTable consists of information of a general nature that are per-job-set, but are
2545         not per-job. See Section 2 entitled 'Terminology and Job Model' for the definition of a job set."
2546     REFERENCE
2547         "The MANDATORY-GROUP macro specifies that this group is MANDATORY."
2548     ::= { jmGeneral 1 }
2549
2550 jmGeneralEntry OBJECT-TYPE
2551     SYNTAX      JmGeneralEntry
2552     MAX-ACCESS  not-accessible
2553     STATUS      current
2554     DESCRIPTION
2555         "Information about a job set (queue).
2556
2557         An entry SHALL exist in this table for each job set."
2558     INDEX { jmGeneralJobSetIndex }
2559     ::= { jmGeneralTable 1 }
2560
2561 JmGeneralEntry ::= SEQUENCE {
2562     jmGeneralJobSetIndex           Integer32(1..32767),
2563     jmGeneralNumberOfActiveJobs   Integer32(0..2147483647),
2564     jmGeneralOldestActiveJobIndex Integer32(0..2147483647),
2565     jmGeneralNewestActiveJobIndex Integer32(0..2147483647),
2566     jmGeneralJobPersistence       Integer32(15..2147483647),
2567     jmGeneralAttributePersistence Integer32(15..2147483647),
2568     jmGeneralJobSetName           OCTET STRING(SIZE(0..63))
2569 }
2570
2571 jmGeneralJobSetIndex OBJECT-TYPE
2572     SYNTAX      Integer32(1..32767)
2573     MAX-ACCESS  not-accessible
2574     STATUS      current
2575     DESCRIPTION
2576         "A unique value for each job set in this MIB. The jmJobTable and jmAttributeTable tables
2577         have this same index as their primary index.
2578

```

2579 The value(s) of the **jmGeneralJobSetIndex** SHALL be persistent across power cycles, so that  
 2580 clients that have retained **jmGeneralJobSetIndex** values will access the same job sets upon  
 2581 subsequent power-up.  
 2582

2583 An implementation that has only one job set, such as a printer with a single queue, SHALL hard  
 2584 code this object with the value **1**."

2585 REFERENCE  
 2586 "See Section 2 entitled 'Terminology and Job Model' for the definition of a job set.  
 2587 Corresponds to the first index in **jmJobTable** and **jmAttributeTable**."  
 2588 ::= { jmGeneralEntry 1 }  
 2589

2590 **jmGeneralNumberOfActiveJobs** OBJECT-TYPE  
 2591 SYNTAX Integer32(0..2147483647)  
 2592 MAX-ACCESS read-only  
 2593 STATUS current  
 2594 DESCRIPTION  
 2595 "The current number of 'active' jobs in the **jmJobIDTable**, **jmJobTable**, and  
 2596 **jmAttributeTable**, i.e., the total number of jobs that are in the **pending**, **processing**, or  
 2597 **processingStopped** states. See the **JmJobStateTC** textual-convention for the exact  
 2598 specification of the semantics of the job states."  
 2599 ::= { jmGeneralEntry 2 }  
 2600

2601 **jmGeneralOldestActiveJobIndex** OBJECT-TYPE  
 2602 SYNTAX Integer32 (0..2147483647)  
 2603 MAX-ACCESS read-only  
 2604 STATUS current  
 2605 DESCRIPTION  
 2606 "The **jmJobIndex** of the oldest job that is still in one of the 'active' states (**pending**, **processing**,  
 2607 or **processingStopped**). In other words, the index of the 'active' job that has been in the job  
 2608 tables the longest.  
 2609  
 2610 If there are no active jobs, the agent SHALL set the value of this object to **0**."  
 2611 REFERENCE  
 2612 "See Section 3.2 entitled 'The Job Tables and the Oldest Active and Newest Active Indexes' for  
 2613 a description of the usage of this object."  
 2614 ::= { jmGeneralEntry 3 }  
 2615

2616 **jmGeneralNewestActiveJobIndex** OBJECT-TYPE  
 2617 SYNTAX Integer32 (0..2147483647)  
 2618 MAX-ACCESS read-only  
 2619 STATUS current  
 2620 DESCRIPTION  
 2621 "The **jmJobIndex** of the newest job that is in one of the 'active' states (**pending**, **processing**, or  
 2622 **processingStopped**). In other words, the index of the 'active' job that has been most recently  
 2623 added to the **job tables**.  
 2624  
 2625 When all jobs become 'inactive', i.e., enter the **pendingHeld**, **completed**, **canceled**, or **aborted**  
 2626 states, the agent SHALL set the value of this object to **0**."  
 2627 REFERENCE

2628 "See Section 3.2 entitled 'The Job Tables and the Oldest Active and Newest Active Indexes' for  
 2629 a description of the usage of this object."  
 2630 ::= { jmGeneralEntry 4 }  
 2631

**jmGeneralJobPersistence** OBJECT-TYPE  
 2632 SYNTAX **Integer32(15..2147483647)**  
 2633 UNITS "seconds"  
 2634 MAX-ACCESS read-only  
 2635 STATUS current  
 2636 DESCRIPTION  
 2637 "The minimum time in seconds for this instance of the Job Set that an entry SHALL remain in  
 2638 the **jmJobIDTable** and **jmJobTable** after **processing** has *completed*, i.e., the minimum time in  
 2639 seconds starting when the job enters the **completed, canceled, or aborted** state.  
 2640  
 2641 Depending on implementation, the value of this object MAY be either: (1) set by the system  
 2642 administrator by means outside this specification or (2) fixed by the implementation.  
 2643  
 2644 This value SHALL be equal to or greater than the value of **jmGeneralAttributePersistence**.  
 2645 This value SHOULD be at least 60 which gives a monitoring application one minute in which to  
 2646 poll for job data."  
 2647 DEFVAL { 60 } -- one minute  
 2648 ::= { jmGeneralEntry 5 }  
 2649

**jmGeneralAttributePersistence** OBJECT-TYPE  
 2650 SYNTAX **Integer32(15..2147483647)**  
 2651 UNITS "seconds"  
 2652 MAX-ACCESS read-only  
 2653 STATUS current  
 2654 DESCRIPTION  
 2655 "The minimum time in seconds for this instance of the Job Set that an entry SHALL remain in  
 2656 the **jmAttributeTable** after **processing** has *completed*, i.e., the time in seconds starting when  
 2657 the job enters the **completed, canceled, or aborted** state.  
 2658  
 2659 Depending on implementation, the value of this object MAY be either (1) set by the system  
 2660 administrator by means outside this specification or MAY be (2) fixed by the implementation.  
 2661  
 2662 This value SHOULD be at least 60 which gives a monitoring application one minute in which to  
 2663 poll for job data."  
 2664 DEFVAL { 60 } -- one minute  
 2665 ::= { jmGeneralEntry 6 }  
 2666

**jmGeneralJobSetName** OBJECT-TYPE  
 2667 SYNTAX **OCTET STRING(SIZE(0..63))**  
 2668 MAX-ACCESS read-only  
 2669 STATUS current  
 2670 DESCRIPTION  
 2671 "The human readable name of this job set assigned by the system administrator (by means  
 2672 outside of this MIB). Typically, this name SHOULD be the name of the job queue. If a server  
 2673 or device has only a single job set, this object can be the administratively assigned name of the  
 2674  
 2675  
 2676

2677 server or device itself. This name does not need to be unique, though each job set in a single  
 2678 Job Monitoring MIB SHOULD have distinct names.  
 2679

2680 NOTE - The purpose of this object is to help the user of the job monitoring application  
 2681 distinguish between several job sets in implementations that support more than one job set."  
 2682 REFERENCE  
 2683 "See the OBJECT compliance macro for the minimum maximum length required for  
 2684 conformance."  
 2685 ::= { jmGeneralEntry 7 }  
 2686  
 2687  
 2688  
 2689  
 2690 -- The Job ID Group (MANDATORY)  
 2691  
 2692 -- The **jmJobIDGroup** consists entirely of the **jmJobIDTable**.  
 2693  
 2694  
 2695 jmJobID OBJECT IDENTIFIER ::= { jobmonMIBObjects 2 }  
 2696  
 2697 jmJobIDTable OBJECT-TYPE  
 2698 SYNTAX SEQUENCE OF JmJobIDEntry  
 2699 MAX-ACCESS not-accessible  
 2700 STATUS current  
 2701 DESCRIPTION  
 2702 "The **jmJobIDTable** provides a correspondence map (1) between the job submission ID that a  
 2703 client uses to refer to a job and (2) the **jmGeneralJobSetIndex** and **jmJobIndex** that the Job  
 2704 Monitoring MIB agent assigned to the job and that are used to access the job in all of the other  
 2705 tables in the MIB. If a monitoring application already knows the **jmGeneralJobSetIndex** and  
 2706 the **jmJobIndex** of the job it is querying, that application NEED NOT use the **jmJobIDTable**."  
 2707 REFERENCE  
 2708 "The MANDATORY-GROUP macro specifies that this group is MANDATORY."  
 2709 ::= { jmJobID 1 }  
 2710  
 2711 jmJobIDEntry OBJECT-TYPE  
 2712 SYNTAX JmJobIDEntry  
 2713 MAX-ACCESS not-accessible  
 2714 STATUS current  
 2715 DESCRIPTION  
 2716 "The map from (1) the **jmJobSubmissionID** to (2) the **jmGeneralJobSetIndex** and  
 2717 **jmJobIndex**.  
 2718  
 2719 An entry SHALL exist in this table for each job currently known to the agent for all job sets and  
 2720 job states. Each job SHALL appear in one and only one job set."  
 2721 INDEX { **jmJobSubmissionID** }  
 2722 ::= { jmJobIDTable 1 }  
 2723  
 2724 JmJobIDEntry ::= SEQUENCE {  
 2725 **jmJobSubmissionID** OCTET STRING(SIZE(48)),

```

2726     jmJobIDJobSetIndex                               Integer32(1..32767),
2727     jmJobIDJobIndex                                   Integer32(1..2147483647)
2728 }
2729
2730 jmJobSubmissionID OBJECT-TYPE
2731     SYNTAX      OCTET STRING(SIZE(48))
2732     MAX-ACCESS  not-accessible
2733     STATUS      current
2734     DESCRIPTION
2735         "A quasi-unique 48-octet fixed-length string ID which identifies the job within a particular
2736         client-server environment. There are multiple formats for the jmJobSubmissionID. See the
2737         JmJobSubmissionIDTypeTC textual convention. Each format SHALL be registered using the
2738         procedures of a type 2 enum. See section 3.6.3 entitled: 'IANA Registration of Job Submission
2739         Id Formats'.
2740
2741         If the requester (client or server) does not supply a job submission ID in the job submission
2742         protocol, then the recipient (server or device) SHALL assign a job submission ID using any of
2743         the standard formats and adding the final 8 octets to distinguish the ID from others submitted
2744         from the same requester.
2745
2746         The monitoring application, whether in the client or running separately, MAY use the job
2747         submission ID to help identify which jmJobIndex was assigned by the agent, i.e., in which row
2748         the job information is in the other tables.
2749
2750         NOTE - fixed-length is used so that a management application can use a shortened GetNext
2751         varbind (in SNMPv1 and SNMPv2) in order to get the next submission ID, disregarding the
2752         remainder of the ID in order to access jobs independent of the trailing identifier part, e.g., to get
2753         all jobs submitted by a particular jmJobOwner or from a particular MAC address."
2754     ::= { jmJobIDEntry 1 }
2755
2756 jmJobIDJobSetIndex OBJECT-TYPE
2757     SYNTAX      Integer32(1..32767)
2758     MAX-ACCESS  read-only
2759     STATUS      current
2760     DESCRIPTION
2761         "This object contains the value of the jmGeneralJobSetIndex for the job with the
2762         jmJobSubmissionID value, i.e., the job set index of the job set in which the job was placed
2763         when that server or device accepted the job. This 16-bit value in combination with the
2764         jmJobIDJobIndex value permits the management application to access the other tables to
2765         obtain the job-specific objects for this job."
2766     REFERENCE
2767         "See jmGeneralJobSetIndex in the jmGeneralTable."
2768     ::= { jmJobIDEntry 2 }
2769
2770 jmJobIDJobIndex OBJECT-TYPE
2771     SYNTAX      Integer32(1..2147483647)
2772     MAX-ACCESS  read-only
2773     STATUS      current
2774     DESCRIPTION

```

```

2775         "This object contains the value of the jmJobIndex for the job with the jmJobSubmissionID
2776         value, i.e., the job index for the job when the server or device accepted the job. This value, in
2777         combination with the jmJobIDJobSetIndex value, permits the management application to
2778         access the other tables to obtain the job-specific objects for this job."
2779     REFERENCE
2780         "See jmJobIndex in the jmJobTable."
2781     ::= { jmJobIDEntry 3 }
2782
2783
2784
2785
2786 -- The Job Group (MANDATORY)
2787
2788 -- The jmJobGroup consists entirely of the jmJobTable.
2789
2790 jmJob OBJECT IDENTIFIER ::= { jobmonMIBObjects 3 }
2791
2792 jmJobTable OBJECT-TYPE
2793     SYNTAX     SEQUENCE OF JmJobEntry
2794     MAX-ACCESS not-accessible
2795     STATUS     current
2796     DESCRIPTION
2797         "The jmJobTable consists of basic job state and status information for each job in a job set that
2798         (1) monitoring applications need to be able to access in a single SNMP Get operation, (2) that
2799         have a single value per job, and (3) that SHALL always be implemented."
2800     REFERENCE
2801         "The MANDATORY-GROUP macro specifies that this group is MANDATORY."
2802     ::= { jmJob 1 }
2803
2804 jmJobEntry OBJECT-TYPE
2805     SYNTAX     JmJobEntry
2806     MAX-ACCESS not-accessible
2807     STATUS     current
2808     DESCRIPTION
2809         "Basic per-job state and status information.
2810
2811         An entry SHALL exist in this table for each job, no matter what the state of the job is. Each job
2812         SHALL appear in one and only one job set."
2813     REFERENCE
2814         "See Section 3.2 entitled 'The Job Tables'."
2815     INDEX { jmGeneralJobSetIndex, jmJobIndex }
2816     ::= { jmJobTable 1 }
2817
2818 JmJobEntry ::= SEQUENCE {
2819     jmJobIndex                Integer32(1..2147483647),
2820     jmJobState                JmJobStateTC,
2821     jmJobStateReasons1       JmJobStateReasons1TC,
2822     jmNumberOfInterveningJobs Integer32(-2..2147483647),
2823     jmJobKOctetsRequested    Integer32(-2..2147483647),

```

```

2824     jmJobKOctetsProcessed          Integer32(-2..2147483647),
2825     jmJobImpressionsRequested      Integer32(-2..2147483647),
2826     jmJobImpressionsCompleted     Integer32(-2..2147483647),
2827     jmJobOwner                     OCTET STRING(SIZE(0..63))
2828 }
2829
2830 jmJobIndex OBJECT-TYPE
2831     SYNTAX      Integer32(1..2147483647)
2832     MAX-ACCESS  not-accessible
2833     STATUS      current
2834     DESCRIPTION
2835         "The sequential, monotonically increasing identifier index for the job generated by the server or
2836         device when that server or device accepted the job. This index value permits the management
2837         application to access the other tables to obtain the job-specific row entries.
2838
2839         Agents providing access to systems that contain jobs with a job identifier of 0 SHALL map the
2840         job identifier value 0 to a jmJobIndex value that is one higher than the highest job identifier
2841         value that any job can have on that system."
2842     REFERENCE
2843         "See Section 3.2 entitled 'The Job Tables'."
2844         See also jmGeneralNewestActiveJobIndex for the largest value of jmJobIndex.
2845         See JmJobSubmissionTypeTC for a limit on the size of this index if the agent represents it as
2846         an 8-digit decimal number."
2847     ::= { jmJobEntry 1 }
2848
2849 jmJobState OBJECT-TYPE
2850     SYNTAX      JmJobStateTC
2851     MAX-ACCESS  read-only
2852     STATUS      current
2853     DESCRIPTION
2854         "The current state of the job (pending, processing, completed, etc.). Agents SHALL
2855         implement only those states which are appropriate for the particular implementation. However,
2856         management applications SHALL be prepared to receive all the standard job states.
2857
2858         The final value for this object SHALL be one of: completed, canceled, or aborted. The
2859         minimum length of time that the agent SHALL maintain MIB data for a job in the completed,
2860         canceled, or aborted state before removing the job data from the jmJobIDTable and
2861         jmJobTable is specified by the value of the jmGeneralJobPersistence object."
2862     ::= { jmJobEntry 2 }
2863
2864 jmJobStateReasons1 OBJECT-TYPE
2865     SYNTAX      JmJobStateReasons1TC
2866     MAX-ACCESS  read-only
2867     STATUS      current
2868     DESCRIPTION
2869         "Additional information about the job's current state, i.e., information that augments the value of
2870         the job's jmJobState object.
2871

```

2872 Implementation of any reason values is OPTIONAL, but an agent SHOULD return any reason  
 2873 information available. These values MAY be used with any job state or states for which the  
 2874 reason makes sense. Furthermore, when implemented as with any MIB data, the agent SHALL  
 2875 return these values when the reason applies and SHALL NOT return them when the reason no  
 2876 longer applies whether the value of the job's **jmJobState** object changed or not. When the  
 2877 agent cannot provide a reason for the current state of the job~~job does not have any reasons for~~  
 2878 ~~being in its current state~~, the agent SHALL set the value of the **jmJobStateReasons1** object and  
 2879 **jobStateReasonsN** attributes to 0."

## REFERENCE

2881 "The **jobStateReasonsN** ( $N=2..4$ ) attributes provide further additional information about the  
 2882 job's current state."  
 2883 ::= { jmJobEntry 3 }

**jmNumberOfInterveningJobs** OBJECT-TYPE

2886 SYNTAX Integer32(-2..2147483647)

2887 MAX-ACCESS read-only

2888 STATUS current

2889 DESCRIPTION

2890 "The number of jobs that are expected to be processed *before* this job is processed according to  
 2891 the implementation's queuing algorithm if no other jobs were to be submitted. In other words,  
 2892 this value is the job's queue position. The agent SHALL return a value of 0 for this attribute  
 2893 while the job is processing."

2894 ::= { jmJobEntry 4 }

**jmJobKOctetsRequested** OBJECT-TYPE

2897 SYNTAX Integer32(-2..2147483647)

2898 MAX-ACCESS read-only

2899 STATUS current

2900 DESCRIPTION

2901 "The total size in K (1024) octets of the document(s) being requested to be processed in the job.  
 2902 The agent SHALL round the actual number of octets up to the next highest K. Thus 0 octets  
 2903 SHALL be represented as '0', 1-1024 octets SHALL be represented as '1', 1025-2048 SHALL  
 2904 be represented as '2', etc.

2906 In computing this value, the server/device SHALL *not* include the multiplicative factors  
 2907 contributed by (1) the number of document copies, and (2) the number of job copies,  
 2908 independent of whether the device can process multiple copies of the job or document without  
 2909 making multiple passes over the job or document data and independent of whether the output is  
 2910 collated or not. Thus the server/device computation is independent of the implementation."

2911 ::= { jmJobEntry 5 }

**jmJobKOctetsProcessed** OBJECT-TYPE

2914 SYNTAX Integer32(-2..2147483647)

2915 MAX-ACCESS read-only

2916 STATUS current

2917 DESCRIPTION

2918 "The current number of octets processed by the server or device measured in units of K (1024)  
 2919 octets. The agent SHALL round the actual number of octets processed up to the next higher K.  
 2920 Thus 0 octets SHALL be represented as '0', 1-1024 octets SHALL be represented as '1', 1025-



2921 2048 octets SHALL be '2', etc. For printing devices, this value is the number interpreted by the  
 2922 page description language interpreter rather than what has been marked on media.  
 2923

2924 For implementations where multiple copies are produced by the interpreter with only a single  
 2925 pass over the data, the final value SHALL be equal to the value of the  
 2926 **jmJobKOctetsRequested** object. For implementations where multiple copies are produced by  
 2927 the interpreter by processing the data for each copy, the final value SHALL be a multiple of the  
 2928 value of the **jmJobKOctetsRequested** object.  
 2929

2930 NOTE - See the **impressionsCompletedCurrentCopy** and **pagesCompletedCurrentCopy**  
 2931 attributes for attributes that are reset on each document copy.  
 2932

2933 NOTE - The **jmJobKOctetsProcessed** object can be used with the **jmJobKOctetsRequested**  
 2934 object to provide an indication of the relative progress of the job, provided that the  
 2935 multiplicative factor is taken into account for some implementations of multiple copies."  
 2936 ::= { jmJobEntry 6 }  
 2937

2938 **jmJobImpressionsRequested** OBJECT-TYPE  
 2939 SYNTAX **Integer32(-2..2147483647)**  
 2940 MAX-ACCESS read-only  
 2941 STATUS current  
 2942 DESCRIPTION  
 2943 "The number of impressions requested by this job to produce."  
 2944 ::= { jmJobEntry 7 }  
 2945

2946 **jmJobImpressionsCompleted** OBJECT-TYPE  
 2947 SYNTAX **Integer32(-2..2147483647)**  
 2948 MAX-ACCESS read-only  
 2949 STATUS current  
 2950 DESCRIPTION  
 2951 "The current number of impressions completed for this job so far. For printing devices, the  
 2952 impressions completed includes interpreting, marking, and stacking the output. For other types  
 2953 of job services, the number of impressions completed includes the number of impressions  
 2954 processed."  
 2955 ::= { jmJobEntry 8 }  
 2956

2957 **jmJobOwner** OBJECT-TYPE  
 2958 SYNTAX **OCTET STRING(SIZE(0..63))**  
 2959 MAX-ACCESS read-only  
 2960 STATUS current  
 2961 DESCRIPTION  
 2962 "The coded character set name of the user that submitted the job. The method of assigning this  
 2963 user name will be system and/or site specific but the method MUST insure that the name is  
 2964 unique to the network that is visible to the client and target device.  
 2965  
 2966 This value SHOULD be the *authenticated* name of the user submitting the job."  
 2967 REFERENCE  
 2968 "See the OBJECT compliance macro for the minimum maximum length required for  
 2969 conformance."

```

2970 ::= { jmJobEntry 9 }
2971
2972
2973
2974
2975 -- The Attribute Group (MANDATORY)
2976
2977 -- The jmAttributeGroup consists entirely of the jmAttributeTable.
2978 --
2979 -- Implementation of the two objects in this group is MANDATORY.
2980 -- See Section 3.1 entitled 'Conformance Considerations'.
2981 -- An agent SHALL implement any attribute if (1) the server or device
2982 -- supports the functionality represented by the attribute and (2) the
2983 -- information is available to the agent.
2984
2985 jmAttribute OBJECT IDENTIFIER ::= { jobmonMIBObjects 4 }
2986
2987 jmAttributeTable OBJECT-TYPE
2988     SYNTAX      SEQUENCE OF JmAttributeEntry
2989     MAX-ACCESS  not-accessible
2990     STATUS      current
2991     DESCRIPTION
2992         "The jmAttributeTable SHALL contain attributes of the job and document(s) for each job in a
2993         job set. Instead of allocating distinct objects for each attribute, each attribute is represented as a
2994         separate row in the jmAttributeTable."
2995     REFERENCE
2996         "The MANDATORY-GROUP macro specifies that this group is MANDATORY. An agent
2997         SHALL implement any attribute if (1) the server or device supports the functionality represented
2998         by the attribute and (2) the information is available to the agent. "
2999     ::= { jmAttribute 1 }
3000
3001 jmAttributeEntry OBJECT-TYPE
3002     SYNTAX      JmAttributeEntry
3003     MAX-ACCESS  not-accessible
3004     STATUS      current
3005     DESCRIPTION
3006         "Attributes representing information about the job and document(s) or resources required and/or
3007         consumed.
3008
3009         Each entry in the jmAttributeTable is a per-job entry with an extra index for each type of
3010         attribute (jmAttributeTypeIndex) that a job can have and an additional index
3011         (jmAttributeInstanceIndex) for those attributes that can have multiple instances per job. The
3012         jmAttributeTypeIndex object SHALL contain an enum type that indicates the type of attribute
3013         (see the JmAttributeTypeTC textual-convention). The value of the attribute SHALL be
3014         represented in either the jmAttributeValueAsInteger or jmAttributeValueAsOctets objects,
3015         and/or both, as specified in the JmAttributeTypeTC textual-convention.
3016
3017         The agent SHALL create rows in the jmAttributeTable as the server or device is able to
3018         discover the attributes either from the job submission protocol itself or from the document PDL.

```

3019 As the documents are interpreted, the interpreter MAY discover additional attributes and so the  
3020 agent adds additional rows to this table. As the attributes that represent resources are actually  
3021 consumed, the usage counter contained in the **jmAttributeValueAsInteger** object is  
3022 incremented according to the units indicated in the description of the **JmAttributeTypeTC**  
3023 enum.  
3024

3025 The agent SHALL maintain each row in the **jmJobTable** for at least the minimum time after a  
3026 job completes as specified by the **jmGeneralAttributePersistence** object.  
3027

3028 Zero or more entries SHALL exist in this table for each job in a job set."  
3029 REFERENCE  
3030 "See Section 3.3 entitled 'The Attribute Mechanism' for a description of the **jmAttributeTable**."  
3031 INDEX { **jmGeneralJobSetIndex**, **jmJobIndex**, **jmAttributeTypeIndex**,  
3032 **jmAttributeInstanceIndex** }  
3033 ::= { jmAttributeTable 1 }  
3034

3035 **JmAttributeEntry** ::= SEQUENCE {  
3036 **jmAttributeTypeIndex** **JmAttributeTypeTC**,  
3037 **jmAttributeInstanceIndex** **Integer32(1..32767)**,  
3038 **jmAttributeValueAsInteger** **Integer32(-2..2147483647)**,  
3039 **jmAttributeValueAsOctets** **OCTET STRING(SIZE(0..63))**  
3040 }  
3041

3042 **jmAttributeTypeIndex** OBJECT-TYPE  
3043 SYNTAX **JmAttributeTypeTC**  
3044 MAX-ACCESS not-accessible  
3045 STATUS current  
3046 DESCRIPTION  
3047 "The type of attribute that this row entry represents.  
3048  
3049 The type MAY identify information about the job or document(s) or MAY identify a resource  
3050 required to process the job before the job start processing and/or consumed by the job as the job  
3051 is processed.  
3052  
3053 Examples of job and document attributes include: **jobCopiesRequested**,  
3054 **documentCopiesRequested**, **jobCopiesCompleted**, **documentCopiesCompleted**, **fileName**,  
3055 and **documentName**.  
3056  
3057 Examples of required and consumed resource attributes include: **pagesRequested**,  
3058 **pagesCompleted**, **mediumRequested**, and **mediumConsumed**, respectively."  
3059 ::= { jmAttributeEntry 1 }  
3060

3061 **jmAttributeInstanceIndex** OBJECT-TYPE  
3062 SYNTAX **Integer32(1..32767)**  
3063 MAX-ACCESS not-accessible  
3064 STATUS current  
3065 DESCRIPTION  
3066 "A running 16-bit index of the attributes of the same type for each job. For those attributes with  
3067 only a single instance per job, this index value SHALL be 1. For those attributes that are a

3068 single value per document, the index value SHALL be the document number, starting with 1 for  
 3069 the first document in the job. Jobs with only a single document SHALL use the index value of  
 3070 1. For those attributes that can have multiple values per job or per document, such as  
 3071 **documentFormatIndex(37)** or **documentFormat(38)**, the index SHALL be a running index  
 3072 for the job as a whole, starting at 1."  
 3073 ::= { jmAttributeEntry 2 }  
 3074

3075 **jmAttributeValueAsInteger** OBJECT-TYPE  
 3076 SYNTAX **Integer32(-2..2147483647)**  
 3077 MAX-ACCESS read-only  
 3078 STATUS current  
 3079 DESCRIPTION  
 3080 "The integer value of the attribute. The value of the attribute SHALL be represented as an  
 3081 integer if the enum description in the **JmAttributeTypeTC** textual-convention definition has the  
 3082 tag: 'INTEGER:'.  
 3083  
 3084 Depending on the enum definition, this object value MAY be an integer, a counter, an index, or  
 3085 an enum, depending on the **jmAttributeTypeIndex** value. The units of this value are specified  
 3086 in the enum description.  
 3087  
 3088 For those attributes that are accumulating job consumption as the job is processed as specified in  
 3089 the **JmAttributeTypeTC** textual-convention, SHALL contain the final value after the job  
 3090 completes processing, i.e., this value SHALL indicate the total usage of this resource made by  
 3091 the job.  
 3092  
 3093 A monitoring application is able to copy this value to a suitable longer term storage for later  
 3094 processing as part of an accounting system.  
 3095  
 3096 Since the agent MAY add attributes representing resources to this table while the job is waiting  
 3097 to be processed or being processed, which can be a long time before any of the resources are  
 3098 actually used, the agent SHALL set the value of the **jmAttributeValueAsInteger** object to 0  
 3099 for resources that the job has not yet consumed.  
 3100  
 3101 Attributes for which the concept of an integer value is meaningless, such as **fileName**,  
 3102 **interpreter**, and **physicalDevice**, do *not* have the 'INTEGER:' tag in the **JmAttributeTypeTC**  
 3103 definition and so an agent SHALL always return a value of '-1' to indicate 'other' for  
 3104 **jmAttributeValueAsInteger**.  
 3105  
 3106 For attributes which do have the 'INTEGER:' tag in the **JmAttributeTypeTC** definition, if the  
 3107 integer value is not (yet) known, the agent either SHALL not materialize the row in the  
 3108 **jmAttributeTable** until the value is known or SHALL return a '-2' to represent an 'unknown'  
 3109 counting integer value, a '0' to represent an 'unknown' index value, and a '2' to represent an  
 3110 'unknown(2)' enum value."  
 3111 ::= { jmAttributeEntry 3 }  
 3112

3113 **jmAttributeValueAsOctets** OBJECT-TYPE  
 3114 SYNTAX **OCTET STRING(SIZE(0..63))**  
 3115 MAX-ACCESS read-only  
 3116 STATUS current

3117 DESCRIPTION  
3118 "The octet string value of the attribute. The value of the attribute SHALL be represented as an  
3119 OCTET STRING if the enum description in the **JmAttributeTypeTC** textual-convention  
3120 definition has the tag: 'OCTETS:'.  
3121  
3122 Depending on the enum definition, this object value MAY be a coded character set string (text)  
3123 or a binary octet string, such as **DateAndTime**.  
3124  
3125 Attributes for which the concept of an octet string value is meaningless, such as  
3126 **pagesCompleted**, do *not* have the tag 'OCTETS:' in the **JmAttributeTypeTC** definition and so  
3127 the agent SHALL always return a zero length string for the value of the  
3128 **jmAttributeValueAsOctets** object.  
3129  
3130 For attributes which do have the 'OCTETS:' tag in the **JmAttributeTypeTC** definition, if the  
3131 OCTET STRING value is not (yet) known, the agent either SHALL not materialize the row in  
3132 the **jmAttributeTable** until the value is known or SHALL return a zero-length string."  
3133 ::= { jmAttributeEntry 4 }  
3134

```

3135 -- Notifications and Trapping
3136 -- Reserved for the future
3137
3138 jobmonMIBNotifications OBJECT IDENTIFIER ::= { jobmonMIB 2 }
3139
3140
3141
3142 -- Conformance Information
3143
3144 jmMIBConformance OBJECT IDENTIFIER ::= { jobmonMIB 3 }
3145
3146 -- compliance statements
3147 jmMIBCompliance MODULE-COMPLIANCE
3148     STATUS current
3149     DESCRIPTION
3150         "The compliance statement for agents that implement the
3151         job monitoring MIB."
3152     MODULE -- this module
3153     MANDATORY-GROUPS {
3154         jmGeneralGroup, jmJobIDGroup, jmJobGroup, jmAttributeGroup }
3155
3156     OBJECT jmGeneralJobSetName
3157     SYNTAX OCTET STRING (SIZE(0..8))
3158     DESCRIPTION
3159         "Only 8 octets maximum string length NEED be supported by the agent."
3160
3161     OBJECT jmJobOwner
3162     SYNTAX OCTET STRING (SIZE(0..16))
3163     DESCRIPTION
3164         "Only 16 octets maximum string length NEED be supported by the agent."
3165
3166 -- There are no CONDITIONALLY MANDATORY or OPTIONAL groups.
3167
3168     ::= { jmMIBConformance 1 }
3169
3170 jmMIBGroups OBJECT IDENTIFIER ::= { jmMIBConformance 2 }
3171
3172 jmGeneralGroup OBJECT-GROUP
3173     OBJECTS {
3174         jmGeneralNumberOfActiveJobs, jmGeneralOldestActiveJobIndex,
3175         jmGeneralNewestActiveJobIndex, jmGeneralJobPersistence,
3176         jmGeneralAttributePersistence, jmGeneralJobSetName }
3177     STATUS current
3178     DESCRIPTION
3179         "The general group."
3180     ::= { jmMIBGroups 1 }
3181
3182 jmJobIDGroup OBJECT-GROUP
3183     OBJECTS {

```

```
3184         jmJobIDJobSetIndex, jmJobIDJobIndex }
3185     STATUS current
3186     DESCRIPTION
3187         "The job ID group."
3188     ::= { jmMIBGroups 2 }
3189
3190 jmJobGroup OBJECT-GROUP
3191     OBJECTS {
3192         jmJobState, jmJobStateReasons1, jmNumberOfInterveningJobs,
3193         jmJobKOctetsRequested, jmJobKOctetsProcessed, jmJobImpressionsRequested,
3194         jmJobImpressionsCompleted, jmJobOwner }
3195     STATUS current
3196     DESCRIPTION
3197         "The job group."
3198     ::= { jmMIBGroups 3 }
3199
3200 jmAttributeGroup OBJECT-GROUP
3201     OBJECTS {
3202         jmAttributeValueAsInteger, jmAttributeValueAsOctets }
3203     STATUS current
3204     DESCRIPTION
3205         "The attribute group."
3206     ::= { jmMIBGroups 4 }
3207
3208
3209 END
```

## 3210 **5. Appendix A - Implementing the Job Life Cycle**

3211 The job object has well-defined states and client operations that affect the transition between the  
3212 job states. Internal server and device actions also affect the transitions of the job between the job  
3213 states. These states and transitions are referred to as the job's *life cycle*.

3214 Not all implementations of job submission protocols have all of the states of the job model  
3215 specified here. The job model specified here is intended to be a superset of most implementations.  
3216 It is the purpose of the agent to map the particular implementation's job life cycle onto the one  
3217 specified here. The agent MAY omit any states not implemented. Only the **processing** and  
3218 **completed** states are required to be implemented by an agent. However, a conforming  
3219 management application SHALL be prepared to accept any of the states in the job life cycle  
3220 specified here, so that the management application can interoperate with any conforming agent.

3221 The job states are intended to be user visible. The agent SHALL make these states visible in the  
3222 MIB, but only for the subset of job states that the implementation has. Some implementations  
3223 MAY need to have sub-states of these user-visible states. The **jmJobStateReasons1** object and  
3224 the **jobStateReasonsN** ( $N=2..4$ ) attributes can be used to represent the sub-states of the jobs.

3225 Job states are intended to last a user-visible length of time in most implementations. However,  
3226 some jobs may pass through some states in zero time in some situations and/or in some  
3227 implementations.

3228 The job model does not specify how accounting and auditing is implemented, except to assume  
3229 that accounting and auditing logs are separate from the job life cycle and last longer than job  
3230 entries in the MIB. Jobs in the **completed**, **aborted**, or **canceled** states are not logs, since jobs in  
3231 these states are accessible via SNMP protocol operations and SHALL be removed from the Job  
3232 Monitoring MIB tables after a site-settable or implementation-defined period of time. An  
3233 accounting application MAY copy accounting information incrementally to an accounting log as a  
3234 job processes, or MAY be copied while the job is in the **canceled**, **aborted**, or **completed** states,  
3235 depending on implementation. The same is true for auditing logs.

3236 **The jmJobState object specifies the standard job states. The normal job state transitions**  
3237 **are shown in the state transition diagram presented in Table 1.**

## 3238 **6. APPENDIX B - Support of the Job Submission ID in Job Submission** 3239 **Protocols**

3240 This appendix lists the job submission protocols that support the concept of a job  
3241 submission ID and indicates the attribute used in that job submission protocol.



3242 **6.1 Hewlett-Packard's Printer Job Language (PJL)**

3243 Hewlett-Packard's Printer Job Language provides job-level printer control and printer  
 3244 status information to applications. The PJL JOB command is used at the beginning of a  
 3245 print job and can include options applying only to that job. A PJL JOB command option  
 3246 has been defined to facilitate passing the **JobSubmissionID** with the print job, as required  
 3247 by the Job Monitoring MIB. The option is of the form:

```
3248
3249     SUBMISSIONID = "id string"
3250
```

3251 Where the "id string" is a string and SHALL be enclosed in double quotes. The format is  
 3252 as described for the **jmJobSubmissionID** object.

3253 The entire PJL JOB command with the optional parameter would be of the form:

```
3254
3255     @PJL JOB SUBMISSIONID = "id string"
3256
```

3257 See "Printer Job Language Technical Reference Manual", part number 5021-0328, from  
 3258 Hewlett-Packard for complete information on the PJL JOB command and the Printer Job  
 3259 Language.

3260 **6.2 ISO DPA**

3261 The ISO 10175 Document Printing Application (DPA) protocol specifies the "**job-client-**  
 3262 **id**" attribute that allows the client to supply a text string ID for each job.

3263 **7. References**

3264 [hr-mib] P. Grillo, S. Waldbusser, "Host Resources MIB", RFC 1514, September 1993

3265 [iana] J. Reynolds, and J. Postel, "Assigned Numbers", STD 2, RFC 1700, ISI, October  
 3266 1994.

3267 [iana-media-types] IANA Registration of MIME media types (MIME content  
 3268 types/subtypes). See <ftp://ftp.isi.edu/in-notes/iana/assignments/>

3269 [iso-dpa] ISO/IEC 10175 Document Printing Application (DPA). See  
 3270 <ftp://ftp.pwg.org/pub/pwg/dpa/>

3271 [ipp-model] Internet Printing Protocol (IPP), work in progress on the IETF standards  
 3272 track. See [draft-ietf-ipp-model-01.txt](#). See also <http://www.pwg.org/ipp/index.html>

3273 [mib-II] MIB-II, RFC 1213.

3274 [print-mib] The Printer MIB - RFC 1759, proposed IETF standard. Also an Internet-  
 3275 Draft on the standards track as a draft standard: [draft-ietf-printmib-mib-info-021.txt](#)

- 3276 [req-words] S. Bradner, "Keywords for use in RFCs to Indicate Requirement Levels",  
3277 RFC 2119, March 1997.
- 3278 [rfc 2130] C. Weider, C. Preston, K. Simonsen, H. Alvestrand, R. Atkinson, M. Crispin,  
3279 and P. Svanberg, "The Report of the IAB Character Set Workshop held 29 Feb-1 March,  
3280 1997", April 1997, RFC 2130.
- 3281 [SMIv2-TC] ~~SNMPv2-TC, RFC 1903~~, J. Case, et al. "Textual Conventions for Version 2  
3282 of the Simple Network Management Protocol (SNMPv2)", RFC 1903, January 1996.
- 3283 [tipsi] IEEE 1284.1, Transport-independent Printer System Interface (TIPSI).
- 3284 [URI-spec] Berners-Lee, T., Masinter, L., McCahill, M. , "Uniform Resource Locators  
3285 (URL)", RFC 1738, December, 1994.

## 3286 8. Author's Addresses

- 3287 Ron Bergman  
3288 Dataproducts Corp.  
3289 1757 Tapo Canyon Road  
3290 Simi Valley, CA 93063-3394  
3291  
3292 Phone: 805-578-4421  
3293 Fax: 805-578-4001  
3294 Email: rbergman@dpc.com  
3295  
3296  
3297 Tom Hastings  
3298 Xerox Corporation, ESAE-231  
3299 701 S. Aviation Blvd.  
3300 El Segundo, CA 90245  
3301  
3302 Phone: 310-333-6413  
3303 Fax: 310-333-5514  
3304 EMail: hastings@cp10.es.xerox.com  
3305  
3306  
3307 Scott A. Isaacson  
3308 Novell, Inc.  
3309 122 E 1700 S  
3310 Provo, UT 84606  
3311  
3312 Phone: 801-861-7366  
3313 Fax: 801-861-4025

3314 EMail: scott\_isaacson@novell.com

3315

3316

3317 Harry Lewis

3318 IBM Corporation

3319 6300 Diagonal Hwy

3320 Boulder, CO 80301

3321

3322 Phone: (303) 924-5337

3323 Fax:

3324 Email: harryl@us.ibm.com

3325

3326

3327 Send comments to the printmib WG using the Job Monitoring Project (JMP)

3328 Mailing List: jmp@pwg.org

3329

3330 To learn how to subscribe, send email to: jmp-request@pwg.org

3331

3332 For further information, access the PWG web page under "JMP":

3333 <http://www.pwg.org/>

3334

3335 Other Participants:

3336 Chuck Adams - Tektronix

3337 Jeff Barnett - IBM

3338 Keith Carter, IBM Corporation

3339 Jeff Copeland - QMS

3340 Andy Davidson - Tektronix

3341 Roger deBry - IBM

3342 Mabry Dozier - QMS

3343 Lee Ferrel - Canon

3344 Steve Gebert - IBM

3345 Robert Herriot - Sun Microsystems Inc.

3346 Shige Kanemitsu - Kyocera

3347 David Kellerman - Northlake Software

3348 Rick Landau - Digital

3349 Harry Lewis - IBM

3350 Pete Loya - HP

3351 Ray Lutz - Cognisys

3352 Jay Martin - Underscore

3353 Mike MacKay, Novell, Inc.

3354 Stan McConnell - Xerox  
3355 Carl-Uno Manros, Xerox, Corp.  
3356 Pat Nogay - IBM  
3357 Bob Pentecost - HP  
3358 Rob Rhoads - Intel  
3359 David Roach - Unisys  
3360 Hiroyuki Sato - Canon  
3361 Bob Setterbo - Adobe  
3362 Gail Songer, EFI  
3363 Mike Timperman - Lexmark  
3364 Randy Turner - Sharp  
3365 William Wagner - Digital Products  
3366 Jim Walker - Dazel  
3367 Chris Wellens - Interworking Labs  
3368 Rob Whittle - Novell  
3369 Don Wright - Lexmark  
3370 Lloyd Young - Lexmark  
3371 Atsushi Yuki - Kyocera  
3372 Peter Zehler, Xerox, Corp.

3373 **9. INDEX**

3374 This index includes the textual conventions, the objects, and the attributes. Textual  
 3375 conventions all start with the prefix: "JM" and end with the suffix: "TC". Objects all  
 3376 starts with the prefix: "jm" followed by the group name. Attributes are identified with  
 3377 enums, and so start with any lower case letter and have no special prefix.

		3411	jmGeneralNewestActiveJobIndex .....	64	
3378	—C—	3412	jmGeneralNumberOfActiveJobs .....	64	
		3413	jmGeneralOldestActiveJobIndex .....	64	
3379	colorantConsumed .....	51	3414	jmJobIDJobIndex .....	67
3380	colorantRequested .....	51	3415	jmJobIDJobSetIndex .....	67
		3416	jmJobImpressionsCompleted .....	71	
3381	—D—	3417	jmJobImpressionsRequested .....	71	
		3418	jmJobIndex .....	69	
3382	deviceNameRequested .....	44	3419	jmJobKOctetsProcessed .....	70
3383	documentCopiesCompleted .....	48	3420	jmJobKOctetsRequested .....	70
3384	documentCopiesRequested .....	47	3421	jmJobOwner .....	71
3385	documentFormat .....	45	3422	JmJobServiceTypesTC .....	54
3386	documentFormatIndex .....	45	3423	JmJobSourcePlatformTypeTC .....	30
3387	documentName .....	44	3424	jmJobState .....	69
		3425	jmJobStateReasons1 .....	69	
3388	—F—	3426	JmJobStateReasons1TC .....	55	
		3427	JmJobStateReasons2TC .....	58	
3389	fileName .....	44	3428	JmJobStateReasons3TC .....	61
3390	finishing .....	46	3429	JmJobStateReasons4TC .....	62
3391	fullColorImpressionsCompleted .....	49	3430	JmJobStateTC .....	38
		3431	jmJobSubmissionID .....	67	
3392	—H—	3432	JmJobSubmissionTypeTC .....	36	
		3433	JmMediumTypeTC .....	35	
3393	highlightColorImpressionsCompleted .....	49	3434	jmNumberOfInterveningJobs .....	70
		3435	JmPrinterResolutionTC .....	33	
3394	—I—	3436	JmPrintQualityTC .....	32	
		3437	JmTimeStampTC .....	30	
3395	impressionsCompletedCurrentCopy .....	49	3438	JmTonerEconomyTC .....	34
3396	impressionsInterpreted .....	49	3439	jobAccountName .....	42
3397	impressionsSentToDevice .....	48	3440	jobComment .....	44
3398	impressionsSpooled .....	48	3441	jobCompletedTime .....	52
		3442	jobCopiesCompleted .....	47	
3399	—J—	3443	jobCopiesRequested .....	47	
		3444	jobHold .....	46	
3400	jmAttributeInstanceIndex .....	73	3445	jobHoldUntil .....	46
3401	jmAttributeTypeIndex .....	73	3446	jobKOctetsTransferred .....	48
3402	JmAttributeTypeTC .....	40	3447	jobName .....	42
3403	jmAttributeValueAsInteger .....	74	3448	jobOriginatingHost .....	43
3404	jmAttributeValueAsOctets .....	74	3449	jobPriority .....	45
3405	JmBooleanTC .....	34	3450	jobProcessAfterDateAndTime .....	45
3406	JmFinishingTC .....	31	3451	jobProcessingCPUtime .....	52
3407	jmGeneralAttributePersistence .....	65	3452	jobServiceTypes .....	43
3408	jmGeneralJobPersistence .....	65	3453	jobSourceChannelIndex .....	43
3409	jmGeneralJobSetIndex .....	63	3454	jobSourcePlatformType .....	43
3410	jmGeneralJobSetName .....	65	3455	jobStartedBeingHeldTime .....	52

3456	jobStartedProcessingTime .....	52	3475	printerResolutionRequested .....	47
3457	jobStateReasons2 .....	41	3476	printerResolutionUsed .....	47
3458	jobStateReasons3 .....	41	3477	printQualityRequested .....	46
3459	jobStateReasons4 .....	41	3478	printQualityUsed .....	47
3460	jobSubmissionTime .....	51	3479	processingMessage .....	41
3461	jobSubmissionToServerTime .....	51			
			3480	<b>—Q—</b>	
3462	<b>—M—</b>		3481	queueNameRequested .....	44
3463	mediumConsumed .....	50			
3464	mediumRequested .....	50	3482	<b>—S—</b>	
			3483	serverAssignedJobName .....	42
3465	<b>—N—</b>		3484	sheetsCompleted .....	50
3466	numberOfDocuments .....	44	3485	sheetsCompletedCurrentCopy .....	50
			3486	sheetsRequested .....	50
3467	<b>—O—</b>		3487	sides .....	46
3468	other .....	41	3488	submittingApplicationName .....	43
3469	outputBin .....	46	3489	submittingServerName .....	43
			3490	<b>—T—</b>	
3470	<b>—P—</b>		3491	tonerDensityRequested .....	47
3471	pagesCompleted .....	49	3492	tonerDensityUsed .....	47
3472	pagesCompletedCurrentCopy .....	49	3493	tonerEcomonyRequested .....	47
3473	pagesRequested .....	49	3494	tonerEcomonyUsed .....	47
3474	physicalDevice .....	44			
3495					