

# Job Monitoring MIB, V0.865

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

From: Tom Hastings

Date: ~~09/1908/08/97~~

Version: 0.865

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

Status: ~~Ninth~~<sup>Eighth</sup> draft MIB that incorporates the agreements reached on the DL on issues in V0.85 which was released after the 8/8 meeting and the agreements reached at the JMP meeting on 9/19~~resolutions of issues 110 to 120 from the 8/8 JMP meeting.~~ In addition to the changes listed in Ron's list, the JMP agreed to remove the finishing enums that IPP removed (because of a lack of a coordinate system specification for stapling), add private enum range for attributes to agree with IPP. 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. See the separate issues.doc and .pdf file.

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

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

There are 65 attributes.



22 INTERNET-DRAFT

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

September 19~~August 8~~, 1997

31

**Job Monitoring MIB - V0.865**

**<draft-ietf-printmib-job-monitor-065.txt>**

**Expires Mar 19~~Feb 8~~, 1997**

35

**Status of this Memo**

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

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

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

**Abstract**

48  
49 This Internet-Draft specifies a small set of read-only SNMP MIB objects for (1)  
50 monitoring the status and progress of print jobs (2) obtaining resource  
51 requirements before a job is processed, (3) monitoring resource consumption while  
52 a job is being processed and (4) collecting resource accounting data after the  
53 completion of a job. This MIB is intended to be implemented (1) in a printer or  
54 (2) in a server that supports one or more printers. Use of the object set is not  
55 limited to printing. However, support for services other than printing is outside  
56 the scope of this Job Monitoring MIB. Future extensions to this MIB may include,  
57 but are not limited to, fax machines and scanners.

58

59

**TABLE OF CONTENTS**

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

61     **1.1 Types of Information in the MIB .....9**

62     **1.2 Types of Job Monitoring Applications .....10**

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

64     **2.1 System Configurations for the Job Monitoring MIB .....14**

65         2.1.1 Configuration 1 - client-printer .....14

66         2.1.2 Configuration 2 - client-server-printer - agent in the server .....15

67         2.1.3 Configuration 3 - client-server-printer - client monitors printer agent and server .....16

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

69     **3.1 Conformance Considerations.....18**

70         3.1.1 Conformance Terminology .....18

71         3.1.2 Agent Conformance Requirements .....18

72             3.1.2.1 MIB II System Group objects .....19

73             3.1.2.2 MIB II Interface Group objects .....19

74             3.1.2.3 Printer MIB objects .....19

75         3.1.3 Job Monitoring Application Conformance Requirements .....19

76     **3.2 The Job Tables and the Oldest Active and Newest Active Indexes .....20**

77     **3.3 The Attribute Mechanism.....21**

78         3.3.1 Conformance of Attribute Implementation.....22

79         3.3.2 Useful, 'Unknown', and 'Other' Values for Objects and Attributes .....23

80         3.3.3 Data Sub-types and Attribute Naming Conventions .....23

81         3.3.4 Single-Value (Row) Versus Multi-Value (MULTI-ROW) Attributes.....24

82         3.3.5 Requested Attributes.....24

83         3.3.6 Consumption Attributes.....25

84         3.3.7 Index Value Attributes .....25

85     **3.4 Job Identification .....25**

86     **3.5 Internationalization Considerations .....26**

87         3.5.1 'JmUTF8StringTC' for text generated by the server or device.....25

88         3.5.2 'JmJobStringTC' for text generated by the job submitter.....25

89         3.5.3 'DateAndTime' for representing the date and time .....25

90     **3.6 IANA Considerations.....27**

91         3.6.1 IANA Registration of enums .....28

92	3.6.1.1 Type 1 enumerations.....	28
93	3.6.1.2 Type 2 enumerations.....	28
94	3.6.1.3 Type 3 enumeration.....	29
95	3.6.2 IANA Registration of type 2 bit values.....	29
96	3.6.3 IANA Registration of Job Submission Id Formats.....	29
97	3.6.4 IANA Registration of MIME types/sub-types for document-formats.....	29
98	<b>3.7 Security Considerations.....</b>	<b>29</b>
99	3.7.1 Read-Write objects.....	29
100	3.7.2 Read-Only Objects In Other User's Jobs.....	30
101	<b>3.8 Values for Objects.....</b>	<b>23</b>
102	<b>3.9 Notifications.....</b>	<b>30</b>
103	<b>4. MIB SPECIFICATION.....</b>	<b>30</b>
104	<b>Textual conventions for this MIB module.....</b>	<b>32</b>
105	JmUTF8StringTC.....	33
106	JmJobStringTC.....	33
107	JmTimeStampTC.....	33
108	JmJobSourcePlatformTypeTC.....	33
109	JmFinishingTC.....	34
110	JmPrintQualityTC.....	35
111	JmPrinterResolutionTC.....	36
112	JmTonerEconomyTC.....	36
113	JmBooleanTC.....	36
114	JmMediumTypeTC.....	37
115	JmJobSubmissionIDTypeTC.....	38
116	JmJobStateTC.....	41
117	JmAttributeTypeTC.....	43
118	other (Int32(-2..) and/or Octets63).....	44
119	Job State attributes.....	44
120	jobStateReasons2 (JmJobStateReasons2TC).....	44
121	jobStateReasons3 (JmJobStateReasons3TC).....	44
122	jobStateReasons4 (JmJobStateReasons4TC).....	45
123	processingMessage (UTF8String63).....	45
124	jobCodedCharSet (CodedCharSet).....	45
125	Job Identification attributes.....	45
126	jobURI (Octets(1..255)).....	46
127	jobAccountName (OctetsJobString63).....	46
128	serverAssignedJobName (JobString63).....	46
129	jobName (JobString63).....	46
130	jobServiceTypes (JmJobServiceTypesTC).....	47
131	jobSourceChannelIndex (Int32(0..)).....	47
132	jobSourcePlatformType (JmJobSourcePlatformTypeTC).....	47
133	submittingServerName (JobString63).....	47
134	submittingApplicationName (JobString63).....	47

135	jobOriginatingHost (JobString63) .....	47
136	deviceNameRequested (JobString63).....	47
137	queueNameRequested (JobString63) .....	48
138	physicalDevice (hrDeviceIndex and/or UTF8String63) .....	48
139	numberOfDocuments (Int32(-2..)).....	48
140	fileName (JobString63).....	48
141	documentName (JobString63).....	48
142	jobComment (JobString63).....	48
143	documentFormatIndex (Int32(0..)).....	48
144	documentFormat (PrtInterpreterLangFamilyTC and/or Octets63).....	49
145	Job Parameter attributes.....	49
146	jobPriority (Int32(1..100)).....	49
147	jobProcessAfterDateAndTime (DateAndTime).....	49
148	jobHold (JmBooleanTC).....	50
149	jobHoldUntil (JobString63).....	50
150	outputBin (Int32(0..) and/or JobString63) .....	50
151	sides (Int32(-2..2)).....	50
152	finishing (JmFinishingTC).....	50
153	Image Quality attributes (requested and used).....	50
154	printQualityRequested (JmPrintQualityTC).....	50
155	printQualityUsed (JmPrintQualityTC).....	50
156	printerResolutionRequested (JmPrinterResolutionTC).....	51
157	printerResolutionUsed (JmPrinterResolutionTC).....	51
158	tonerEcomonyRequested (JmTonerEconomyTC).....	51
159	tonerEcomonyUsed (JmTonerEconomyTC).....	51
160	tonerDensityRequested (Int32(-2..100)).....	51
161	tonerDensityUsed (Int32(-2..100)).....	51
162	Job Progress attributes (requested and consumed) .....	51
163	jobCopiesRequested (Int32(-2..)).....	51
164	jobCopiesCompleted (Int32(-2..)).....	51
165	documentCopiesRequested (Int32(-2..)).....	51
166	documentCopiesCompleted (Int32(-2..)).....	52
167	jobKOctetsTransferred (Int32(-2..)).....	52
168	Impression attributes (requested and consumed) .....	52
169	impressionsSpooled (Int32(-2..)).....	52
170	impressionsSentToDevice (Int32(-2..)).....	52
171	impressionsInterpreted (Int32(-2..)) .....	52
172	impressionsCompletedCurrentCopy (Int32(-2..)).....	53
173	fullColorImpressionsCompleted (Int32(-2..)).....	53
174	highlightColorImpressionsCompleted (Int32(-2..)).....	53
175	Page attributes (requested and consumed).....	53
176	pagesRequested (Int32(-2..)) .....	53
177	pagesCompleted (Int32(-2..)) .....	53
178	pagesCompletedCurrentCopy (Int32(-2..)) .....	54
179	Sheet attributes (requested and consumed).....	54
180	sheetsRequested (Int32(-2..)).....	54
181	sheetsCompleted (Int32(-2..)).....	54
182	sheetsCompletedCurrentCopy (Int32(-2..)).....	54
183	Resource attributes (requested and consumed) .....	54

184	mediumRequested (JmMediumTypeTC and/or JobString63) .....	54
185	mediumConsumed (JobString63) .....	55
186	colorantRequested (Int32(-2..) and/or JobString63) .....	55
187	colorantConsumed (Int32(-2..) and/or JobString63) .....	55
188	Time attributes (set by server or device) .....	55
189	jobSubmissionToServerTime (JmTimeStampTC and/or DateAndTime) .....	56
190	jobSubmissionTime (JmTimeStampTC and/or DateAndTime) .....	56
191	jobStartedBeingHeldTime (JmTimeStampTC and/or DateAndTime) .....	56
192	jobStartedProcessingTime (JmTimeStampTC and/or DateAndTime) .....	56
193	jobCompletedTime (JmTimeStampTC and/or DateAndTime) .....	56
194	jobProcessingCPUTime (Int32(-2..)) .....	56
195	JmJobServiceTypesTC .....	58
196	JmJobStateReasons1TC .....	60
197	JmJobStateReasons2TC .....	63
198	JmJobStateReasons3TC .....	67
199	JmJobStateReasons4TC .....	67
200	<b>The General Group (MANDATORY) .....</b>	<b>68</b>
201	jmGeneralJobSetIndex (Int32(1..32767)) .....	68
202	jmGeneralNumberOfActiveJobs (Int32(0..)) .....	69
203	jmGeneralOldestActiveJobIndex (Int32(0..)) .....	69
204	jmGeneralNewestActiveJobIndex (Int32(0..)) .....	69
205	jmGeneralJobPersistence (Int32(15..)) .....	70
206	jmGeneralAttributePersistence (Int32(15..)) .....	70
207	jmGeneralJobSetName (UTF8String63) .....	70
208	<b>The Job ID Group (MANDATORY) .....</b>	<b>71</b>
209	jmJobSubmissionID (OCTET STRING(SIZE(48))) .....	72
210	jmJobIDJobSetIndex (Int32(1..32767)) .....	72
211	jmJobIDJobIndex (Int32(1..)) .....	73
212	<b>The Job Group (MANDATORY) .....</b>	<b>73</b>
213	jmJobIndex (Int32(1..)) .....	74
214	jmJobState (JmJobStateTC) .....	74
215	jmJobStateReasons1 (JmJobStateReasons1TC) .....	74
216	jmNumberOfInterveningJobs (Int32(-2..)) .....	75
217	jmJobKOctetsRequested (Int32(-2..)) .....	75
218	jmJobKOctetsProcessed (Int32(-2..)) .....	75
219	jmJobImpressionsRequested (Int32(-2..)) .....	76
220	jmJobImpressionsCompleted (Int32(-2..)) .....	76
221	jmJobOwner (JobString63) .....	77
222	<b>The Attribute Group (MANDATORY) .....</b>	<b>77</b>
223	jmAttributeTypeIndex (JmAttributeTypeTC) .....	79
224	jmAttributeInstanceIndex (Int32(1..32767)) .....	79
225	jmAttributeValueAsInteger (Int32(-2..)) .....	79
226	jmAttributeValueAsOctets (Octets63) .....	80

227 **5. APPENDIX A - IMPLEMENTING THE JOB LIFE CYCLE..... 84**

228 **6. APPENDIX B - SUPPORT OF THE JOB SUBMISSION ID IN JOB**  
229 **SUBMISSION PROTOCOLS ..... 84**

230 **6.1 Hewlett-Packard's Printer Job Language (PJL) .....85**

231 **6.2 ISO DPA.....85**

232 **7. REFERENCES..... 85**

233 **8. AUTHOR'S ADDRESSES..... 87**

234 **9. INDEX ..... 90**

235



236

## Job Monitoring MIB

### 237 1. Introduction

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

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

#### 258 1.1 Types of Information in the MIB

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

261 User:

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

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

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

267 Provide error and diagnostic information for jobs that did not successfully  
268 complete.

269 Operator:

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

278 Capacity Planner:

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

281 Accountant:

- 282 Provide information to allow the creation of a record of resources consumed and
- 283 printer usage data for charging users or groups for resources consumed.
- 284 Provide information to allow the prediction of consumable usage and resource
- 285 need.

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

291 **1.2 Types of Job Monitoring Applications**

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

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

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

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

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

## 327 2. Terminology and Job Model

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

330 NOTE - Existing systems use conflicting terms, so these terms are drawn from the ISO  
331 10175 Document Printing Application (DPA) standard[iso-dpa]. For example,  
332 PostScript systems use the term *session* for what is called a *job* in this specification and  
333 the term *job* to mean what is called a *document* in this specification. ~~PJL systems use~~  
334 ~~the term *job* to mean what is called a *job* in this specification. PJL also supports~~  
335 ~~multiple *documents* per job, but does not support specifying per document attributes~~  
336 ~~independently for each document.~~

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

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

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

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

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

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

357 Device: Aa hardware entity that (1) interfaces to humans ~~in human-perceptible means,~~  
358 such as a device that produces marks on paper, or scans marks on paper to produce an  
359 electronic representations, (2) accesses digital media, such as or writes CD-ROMs, or (3)  
360 interfaces electronically to another device, such as sends FAX data to another FAX  
361 device.

362 Printer: Aa *device* that puts marks on media.

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

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

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

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

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

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

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

381 | Proxy: An agent that acts as a concentrator for one or more other agents by accepting  
382 | SNMP operations on the behalf of one or more other agents, forwarding them on to those  
383 | other agents, gathering responses from those other agents and returning them to the  
384 | original requesting monitor.

385 | User: Aa person that uses a client or a monitor.

386 | End User: Aa user that uses a client to submit a print job.

387 | System Operator: Aa user that uses a monitor to monitor the system and carries out tasks  
388 | to keep the system running.

389 | System Administrator: Aa user that specifies policy for the system.

390 | Job Instruction: An instruction specifying how, when, or where the job is to be  
391 | processed. Job instructions MAY be passed in the job submission protocol or MAY be  
392 | embedded in the document data or a combination depending on the job submission  
393 | protocol and implementation.

394 | Document Instruction: An instruction specifying how to process the document.  
395 | Document instructions MAY be passed in the job submission protocol separate from the  
396 | actual document data, or MAY be embedded in the document data or a combination,  
397 | depending on the job submission protocol and implementation.

398 | SNMP Information Object: Aa name, value-pair that specifies an action, a status, or a  
399 | condition in an SNMP MIB. Objects are identified in SNMP by an OBJECT  
400 | IDENTIFIER.

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

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

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

415 **2.1 System Configurations for the Job Monitoring MIB**

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

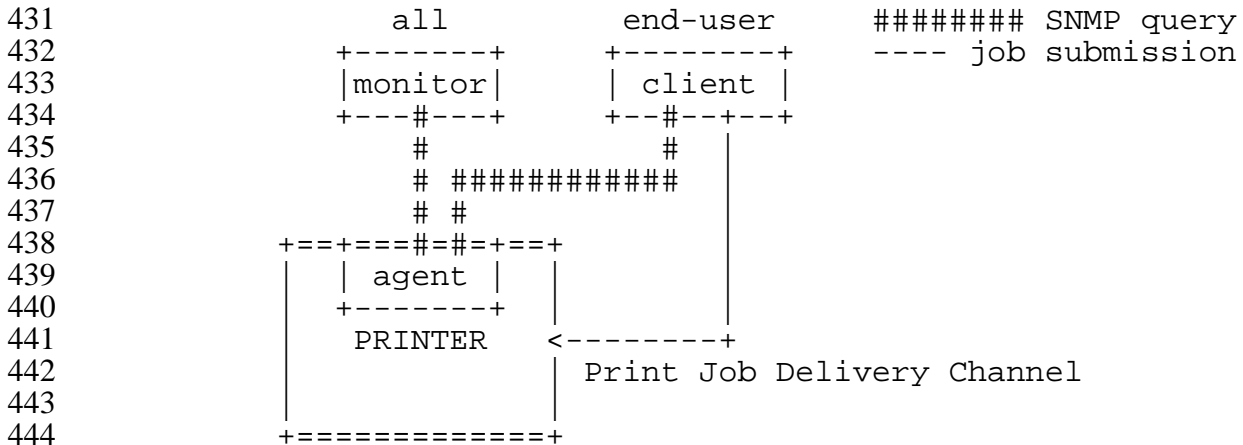
419 The diagram in the Printer MIB[print-mib] entitled: "One Printer's View of the Network"  
 420 is assumed for this MIB as well. Please refer to that diagram to aid in understanding the  
 421 following system configurations.

422 **2.1.1 Configuration 1 - client-printer**

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

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

430



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

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

- 448 1. Multiple **clients** MAY submit jobs to a **printer**.
- 449 2. Multiple **clients** MAY monitor a **printer**.

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

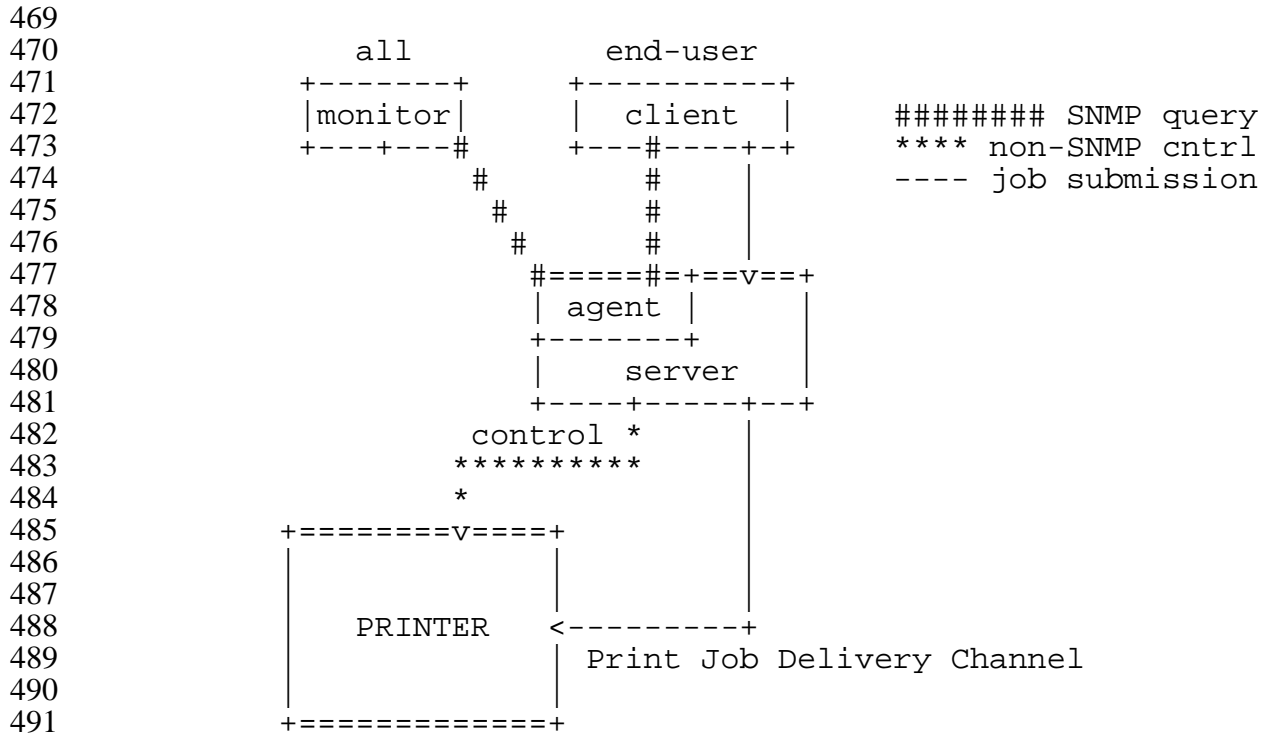
### 453   **2.1.2 Configuration 2 - client-server-printer - agent in the server**

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

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

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

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



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

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

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

502 **2.1.3 Configuration 3 - client-server-printer - client monitors printer agent and**  
503 **server**

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

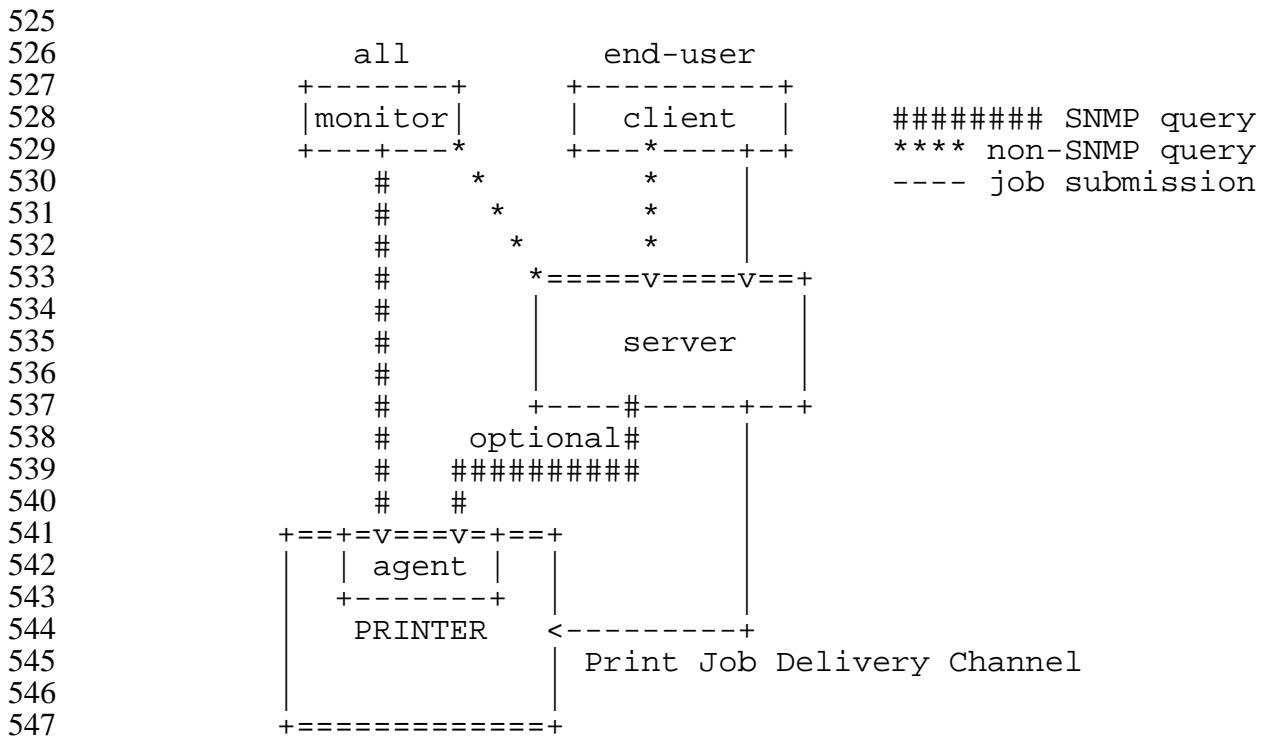
507 The job submitting **client** and/or **monitoring application** monitor jobs by communicating  
508 directly with:

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



- 511 2. A Job Monitoring MIB agent that is part of the **printer** to monitor jobs after  
 512 the **server** passes the jobs to the **printer**. In such configurations, the **server**  
 513 deletes its copy of the job from the **server** after submitting the job to the  
 514 printer usually almost immediately (before the job does much processing, if  
 515 any).

516 In configuration 3, the agent (in the **printer**) SHALL keep the values of the objects in the  
 517 Job Monitoring MIB that the agent implements updated for a job that the server has  
 518 submitted to the printer. The agent SHALL obtain information about the jobs submitted  
 519 to the printer from the server (either in the job submission protocol, in the document data,  
 520 or by direct query of the server), in order to populate some of the objects the Job  
 521 Monitoring MIB in the printer. The agent in the printer SHALL keep the job in the Job  
 522 Monitoring MIB as long as the job is in the Printer, and longer in order to implement the  
 523 **completed** state in which monitoring programs can copy out the accounting data from the  
 524 Job Monitoring MIB.



548 **Figure 2-3 - Configuration 3 - client-server-printer - client monitors printer agent**  
 549 **and server**

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

- 552 1. Multiple **clients** MAY submit jobs to a **server**.  
 553 2. Multiple **clients** MAY monitor a **server**.  
 554 3. Multiple **monitors** MAY monitor a **server**.

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

### 559 3. Managed Object Usage

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

#### 561 3.1 Conformance Considerations

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

##### 565 3.1.1 Conformance Terminology

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

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

##### 579 3.1.2 Agent Conformance Requirements

580 A conforming agent:

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

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

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

#### 592 3.1.2.1 MIB II System Group objects

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

#### 595 3.1.2.2 MIB II Interface Group objects

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

#### 598 3.1.2.3 Printer MIB objects

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

### 605 3.1.3 Job Monitoring Application Conformance Requirements

606 A conforming job monitoring application:

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

### 621 3.2 The Job Tables and the Oldest Active and Newest Active Indexes

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

- 624 1. **jmGeneralJobSetIndex** - which job set
- 625 2. **jmJobIndex** - which job in the job set

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

### 693 3.3 The Attribute Mechanism

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

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

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

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

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

#### 719 3.3.1 Conformance of Attribute Implementation

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

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

### 728 3.3.2 Useful, 'Unknown', and 'Other' Values for Objects and Attributes

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

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

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

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

### 754 3.3.3 Data Sub-types and Attribute Naming Conventions

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

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

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

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

### 770 3.3.4 Single-Value (Row) Versus Multi-Value (MULTI-ROW) Attributes

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

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

### 784 3.3.5 Requested Attributes

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



### 791 3.3.6 Consumption Attributes

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

### 798 3.3.7 Index Value Attributes

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

## 805 3.4 Job Identification

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

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

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

826 ~~it is recommended that the value of the job identifier and the value of jmJobIndex be the~~  
827 ~~same, so that~~

828 The MIB provides a mapping table that maps each **jmJobSubmissionID** value to the  
829 corresponding **jmJobIndex** value generated by the agent, so that an application can  
830 determine the correct value for the **jmJobIndex** value for the job of interest in a single  
831 Get operation, given the Job 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 This section describes the internationalization considerations included in this MIB.

#### 837 3.5.1 Text generated by the server or device

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

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

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

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

#### 853 3.5.2 Text generated by the job submitter

854 All of the objects and attributes represented by the '**JmJobStringTC**' textual-convention  
855 are either (1) supplied in the job submission protocol by the client that submits the job to  
856 the server or device or (2) are defaulted by the server or device if the job submitting client  
857 does not supply values. The agent SHALL represent these objects and attributes in the  
858 MIB either (1) in the coded character set as they were submitted or (2) MAY convert the  
859 coded character set to another coded character set or encoding scheme. In any case, the

860 resulting coded character set representation SHOULD be UTF-8 [UTF-8], but SHALL be  
861 one in which the code positions from 0 to 31 SHALL not be used, 32 to 127 SHALL be  
862 US-ASCII [US-ASCII], 127 SHALL be unused, and the remaining code positions 128 to  
863 255 SHALL represent single-byte or multi-byte graphic characters structured according to  
864 ISO 2022 [ISO 2022] or SHALL be unused.

865 The coded character set SHALL be one of the ones registered with IANA [IANA] and  
866 SHALL be identified by the **jobCodedCharSet** attribute in the **jmJobAttributeTable** for  
867 the job. If the agent does not know what coded character set was used by the job  
868 submitting client, the agent SHALL either (1) return the 'unknown(2)' value for the  
869 **jobCodedCharSet** attribute or (2) not return the **jobCodedCharSet** attribute for the job.

870 Examples of coded character sets which meet this criteria for use as the value of the  
871 **jobCodedCharSet** job attribute are: US-ASCII [US-ASCII], ISO 8859-1 (Latin-1) [ISO  
872 8859-1], any ISO 8859-n, HP Roman8, IBM Code Page 850, Windows Default 8-bit set,  
873 UTF-8 [UTF-8], US-ASCII plus JIS X0208-1990 Japanese [JIS X0208], US-ASCII plus  
874 GB2312-1980 PRC Chinese [GB2312]. See the IANA registry of coded character sets  
875 [IANA charsets].

876 Examples of coded character sets which do not meet this criteria are: national 7-bit sets  
877 conforming to ISO 646 (except US-ASCII), EBCDIC, and ISO 10646 (Unicode) [ISO-  
878 10646]. In order to represent Unicode characters, the UTF-8 [UTF-8] encoding scheme  
879 SHALL be used which has been assigned the MIBenum value of '106' by IANA.

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

### 882 3.5.3 'DateAndTime' for representing the date and time

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

### 886 3.6 IANA Considerations

887 During the development of this standard, the Printer Working Group (PWG) working with  
888 IANA [iana] will register additional enums while the standard is in the proposed and draft  
889 states according to the procedures described in this section. IANA will handle registration  
890 of additional enums after this standard is approved in cooperation with an IANA-  
891 appointed registration editor from the PWG according to the procedures described in this  
892 section:

**893 3.6.1 IANA Registration of enums**

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

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

**905 3.6.1.1 Type 1 enumerations**

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

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

**909 3.6.1.2 Type 2 enumerations**

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

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

- 914 1. JmUTF8StringTC
- 915 2. JmJobStringTC
- 916 3. JmTimeStampTC
- 917 4. JmFinishingTC [same enum values as IPP "finishing" attribute]
- 918 5. JmPrintQualityTC [same enum values as IPP "print-quality" attribute]
- 919 6. JmTonerEconomyTC
- 920 7. JmMediumTypeTC
- 921 8. JmJobSubmissionTypeTC
- 922 9. JmJobStateTC [same enum values as IPP "job-state" attribute]
- 923 10. JmAttributeTypeTC

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

927 3.6.1.3 Type 3 enumeration

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

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

### 932 3.6.2 IANA Registration of type 2 bit values

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

- 934 1. JmJobServiceTypesTC
- 935 2. JmJobStateReasons1TC
- 936 3. JmJobStateReasons2TC
- 937 4. JmJobStateReasons3TC
- 938 5. JmJobStateReasons4TC

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

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

### 944 3.6.3 IANA Registration of Job Submission Id Formats

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

### 949 3.6.4 IANA Registration of MIME types/sub-types for document-formats

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

## 954 3.7 Security Considerations

### 955 3.7.1 Read-Write objects

956 All objects are read-only, greatly simplifying the security considerations. If another MIB  
957 augments this MIB, that MIB might accept SNMP Write operations to objects in that  
958 MIB whose effect is to modify the values of read-only objects in this MIB. However, that

959 MIB SHALL have to support the required access control in order to achieve security, not  
960 this MIB.

961 **3.7.2 Read-Only Objects In Other User's Jobs**

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

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

973 **3.8 Notifications**

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

978 **4. MIB specification**

979 The following pages constitute the actual Job Monitoring MIB.

```

980 Job-Monitoring-MIB DEFINITIONS ::= BEGIN
981
982 IMPORTS
    MODULE-IDENTITY, OBJECT-TYPE, experimental, Integer32
    TEXTUAL-CONVENTION
    MODULE-COMPLIANCE, OBJECT-GROUP
    FROM SNMPv2-SMI
    FROM SNMPv2-TC
    FROM SNMPv2-CONF;

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

983 -- Use the experimental (54) OID assigned to the Printer MIB[print-mib]
984 -- before it was published as RFC 1759.
985 -- Upon publication of the Job Monitoring MIB as an RFC, delete this
986 -- comment and the line following this comment and change the
987 -- reference of { temp 105 } (below) to { mib-2 X }.
988 -- This will result in changing:
989 -- 1 3 6 1 3 54 jobmonMIB(105)  to:
990 -- 1 3 6 1 2 1 jobmonMIB(X)
991 -- This will make it easier to translate prototypes to
992 -- the standard namespace because the lengths of the OIDs won't
993 -- change.
994 temp OBJECT IDENTIFIER ::= { experimental 54 }
995
996 jobmonMIB MODULE-IDENTITY
997 |   LAST-UPDATED "97091908080000Z"
998   ORGANIZATION "IETF Printer MIB Working Group"
999   CONTACT-INFO
1000     "Tom Hastings
1001     Postal: Xerox Corp.
1002     Mail stop ESAE-231
1003     701 S. Aviation Blvd.
1004     El Segundo, CA 90245
1005
1006     Tel: (301)333-6413
1007     Fax: (301)333-5514
1008     E-mail: hastings@cp10.es.xerox.com
1009
1010     Send comments to the printmib WG using the Job Monitoring
1011     Project (JMP) Mailing List: jmp@pwg.org
1012
1013     To learn how to subscribe to the JMP mailing list,
1014     send email to: jmp-request@pwg.org
1015
1016

```

1017 For further information, access the PWG web page under 'JMP':  
 1018 <http://www.pwg.org/>"

1019 DESCRIPTION  
 1020 "The MIB module for monitoring job in servers, printers, and other devices.  
 1021  
 1022 File: draft-ietf-printmib-job-monitor-065.txt  
 1023 Version: 0.865"  
 1024 ::= { temp 105 }  
 1025  
 1026  
 1027  
 1028 -- Textual conventions for this MIB module  
 1029  
 1030  
 1031  
 1032 **JmUTF8StringTC** ::= TEXTUAL-CONVENTION  
 1033 DISPLAY-HINT "255a"  
 1034 STATUS current  
 1035 DESCRIPTION  
 1036 "To facilitate internationalization, this TC represents information taken from the ISO/IEC IS  
 1037 10646-1 character set, encoded as an octet string using the UTF-8 character encoding scheme.  
 1038  
 1039 ~~NOTE—The values of objects and attributes using this textual convention are generated by the~~  
 1040 ~~server or the device, not by the job submitter."~~  
 1041 REFERENCE  
 1042 "See section 3.5.1, entitled: 'Text generated by the server or device'.  
 1043 SYNTAX OCTET STRING (SIZE (0..63))  
 1044  
 1045  
 1046  
 1047  
 1048 **JmJobStringTC** ::= TEXTUAL-CONVENTION  
 1049 STATUS current  
 1050 DESCRIPTION  
 1051 "To facilitate internationalization, this TC represents information using any coded character set  
 1052 registered by IANA as specified in section 3.5.2~~that has the following properties: (1) code~~  
 1053 ~~positions from 0 to 31 SHALL not be used, (2) 32 to 127 SHALL be US ASCII [US ASCII],~~  
 1054 ~~(3) 127 SHALL be unused, and (4) the remaining code positions 128 to 255 SHALL represent~~  
 1055 ~~single byte or multi byte graphic characters structured according to ISO 2022 [ISO 2022] or~~  
 1056 ~~SHALL be unused.~~ While it is recommended that the coded character set be UTF-8 [UTF-8],  
 1057 the actual coded character set SHALL be indicated by the value of the **jobCodedCharSet(7)**  
 1058 attribute for the job.  
 1059  
 1060 ~~NOTE—The values of objects and attributes using this textual convention are either generated~~  
 1061 ~~by the job submitter or defaulted by the server or device when the job submitter does not supply~~  
 1062 ~~values."~~  
 1063 REFERENCE



1064 | "See section 3.5.2, entitled: 'Text generated by the job submitter'."  
 1065 | SYNTAX OCTET STRING (SIZE (0..63))  
 1066 |  
 1067 |  
 1068 |  
 1069 |  
 1070 | **JmTimeStampTC** ::= TEXTUAL-CONVENTION  
 1071 | STATUS current  
 1072 | DESCRIPTION  
 1073 | "The simple time at which an event took place. The units SHALL be in seconds since the  
 1074 | system was booted.  
 1075 |  
 1076 | NOTE - **JmTimeStampTC** is defined in units of seconds, rather than 100ths of seconds, so as  
 1077 | to be simpler for agents to implement (even if they have to implement the 100ths of a second to  
 1078 | comply with implementing **sysUpTime** in MIB-II[mib-II].)  
 1079 |  
 1080 | NOTE - **JmTimeStampTC** is defined as an **Integer32** so that it can be used as a value of an  
 1081 | attribute, i.e., as a value of the **jmAttributeValueAsInteger** object. The **TimeStamp** textual-  
 1082 | convention defined in SMNPv2-TC is defined as an **APPLICATION 3 IMPLICIT INTEGER**  
 1083 | tag, not an **Integer32**, so cannot be used in this MIB as one of the values of  
 1084 | **jmAttributeValueAsInteger**."  
 1085 | SYNTAX INTEGER(0..2147483647)  
 1086 |  
 1087 |  
 1088 |  
 1089 |  
 1090 | **JmJobSourcePlatformTypeTC** ::= TEXTUAL-CONVENTION  
 1091 | STATUS current  
 1092 | DESCRIPTION  
 1093 | "The source platform type that can submit jobs to servers or devices in any of the 3  
 1094 | configurations."  
 1095 | REFERENCE  
 1096 | "This is a type 2 enumeration. See Section 3.6.1.2."  
 1097 | SYNTAX INTEGER {  
 1098 |     **other(1),**  
 1098 |     **unknown(2),**  
 1098 |     **sptUNIX(3),**                     -- UNIX(~~tm~~)  
 1098 |     **sptOS2(4),**                     -- OS/2  
 1098 |     **sptPCDOS(5),**                 -- DOS  
 1098 |     **sptNT(6),**                     -- NT  
 1098 |     **sptMVS(7),**                    -- MVS  
 1098 |     **sptVM(8),**                     -- VM  
 1098 |     **sptOS400(9),**                 -- OS/400  
 1098 |     **sptVMS(10),**                  -- VMS  
 1098 |     **sptWindows95(11),**            -- Windows95  
 1098 |     **sptNetWare(12)**               -- NetWare  
 1098 | }  
 1098 | }

1099  
 1100  
 1101  
 1102  
 1103  
 1104 **JmFinishingTC ::= TEXTUAL-CONVENTION**  
 1105     STATUS     current  
 1106     DESCRIPTION  
 1107         "The type of finishing operation."  
 1108  
 1109         These values are the same as the enum values of the IPP 'finishings' attribute. See Section  
 1110         3.6.1.2.  
 1111  
 1112         **other(1),**  
 1113             Some other finishing operation besides one of the specified or registered values.  
 1114  
 1115         **unknown(2),**  
 1116             The finishing is unknown.  
 1117  
 1118         **none(3),**  
 1119             Perform no finishing.  
 1120  
 1121         **staple(4),**  
 1122             Bind the document(s) with one or more staples. The exact number and placement of the  
 1123             staples is site-defined.  
 1124  
 1125         **stapleTopLeft(5),**  
 1126             Place one or more staples on the top-left corner of the document(s).  
 1127  
 1128         **stapleBottomLeft(6),**  
 1129             Place one or more staples on the bottom-left corner of the document(s).  
 1130  
 1131         **stapleTopRight(7),**  
 1132             Place one or more staples on the top-right corner of the document(s).  
 1133  
 1134         **stapleBottomRight(8),**  
 1135             Place one or more staples on the bottom-right corner of the document(s).  
 1136  
 1137         **saddleStitch(9),**  
 1138             Bind the document(s) with one or more staples (wire stitches) along the middle fold. The  
 1139             exact number and placement of the stitches is site-defined.  
 1140  
 1141         **edgeStitch(10),**  
 1142             Bind the document(s) with one or more staples (wire stitches) along one edge. The exact  
 1143             number and placement of the staples is site-defined.  
 1144  
 1145         **punch(511),**  
 1146             This value indicates that holes are required in the finished document. The exact number

1147 and placement of the holes is site-defined The punch specification MAY be satisfied (in a  
 1148 site- and implementation-specific manner) either by drilling/punching, or by substituting  
 1149 pre-drilled media.  
 1150

**cover(612),**

1151 This value is specified when it is desired to select a non-printed (or pre-printed) cover for  
 1152 the document. This does not supplant the specification of a printed cover (on cover stock  
 1153 medium) by the document itself.  
 1154  
 1155

**bind(713)**

1156 This value indicates that a binding is to be applied to the document; the type and  
 1157 placement of the binding is product-specific."  
 1158

## REFERENCE

1159 "This is a type 2 enumeration. See Section 3.6.1.2."

## SYNTAX INTEGER {

1160 other(1),  
 1161 unknown(2),  
 1162 none(3),  
 1163 staple(4),  
 1164 stapleTopLeft(5),  
 1165 stapleBottomLeft(6),  
 1166 stapleTopRight(7),  
 1167 stapleBottomRight(8),  
 1168 saddleStitch(9),  
 1169 edgeStitch(10),  
 1170 punch(511),  
 1171 cover(612),  
 1172 bind(713)  
 1173 }  
 1174  
 1175  
 1176  
 1177  
 1178  
 1179  
 1180

**JmPrintQualityTC ::= TEXTUAL-CONVENTION**

1181 STATUS current

## DESCRIPTION

1182 "Print quality settings.  
 1183  
 1184  
 1185

1186 These values are the same as the enum values of the IPP 'print-quality' attribute. See Section  
 1187 3.6.1.2."

## REFERENCE

1188 "This is a type 2 enumeration. See Section 3.6.1.2."

## SYNTAX INTEGER {

1189 **other(1),** -- Not one of the specified or registered values.  
 1190 --  
**unknown(2),** -- The actual value is unknown.  
**draft(3),** -- Lowest quality available on the printer.

```

1191     }
1192
1193
1194
1195
1196 JmPrinterResolutionTC ::= TEXTUAL-CONVENTION
1197     STATUS      current
1198     DESCRIPTION
1199         "Printer resolutions.
1200
1201         Nine octets consisting of two 4-octet SIGNED-INTEGERS followed by a SIGNED-BYTE.
1202         The values are the same as those specified in the Printer MIB [printmib]. The first SIGNED-
1203         INTEGER contains the value of prtMarkerAddressabilityXFeedDir. The second SIGNED-
1204         INTEGER contains the value of prtMarkerAddressabilityFeedDir. The SIGNED-BYTE
1205         contains the value of prtMarkerAddressabilityUnit.
1206
1207         Note: the latter value is either 3 (tenThousandsOfInches) or 4 (micrometers) and the
1208         addressability is in 10,000 units of measure. Thus the SIGNED-INTEGERS represent integral
1209         values in either dots-per-inch or dots-per-centimeter.
1210
1211         The syntax is the same as the IPP 'printer-resolution' attribute. See Section 3.6.1.2."
1212     SYNTAX      OCTET STRING (SIZE(9))
1213
1214
1215
1216
1217
1218 JmTonerEconomyTC ::= TEXTUAL-CONVENTION
1219     STATUS      current
1220     DESCRIPTION
1221         "Toner economy settings."
1222     REFERENCE
1223         "This is a type 2 enumeration. See Section 3.6.1.2."
1224     SYNTAX      INTEGER {
1225         unknown(2),      -- unknown.
1226         off(3),          -- Off. Normal. Use full toner.
1227         on(4)           -- On. Use less toner than normal.
1228     }
1229
1230
1231 JmBooleanTC ::= TEXTUAL-CONVENTION

```

```

1232 STATUS current
1233 DESCRIPTION
1234     "Boolean true or false value."
1235 REFERENCE
1236     "This is a type 2 enumeration. See Section 3.6.1.2."
1237 SYNTAX INTEGER {
        unknown(2),      -- unknown.
        false(3),       -- FALSE.
        true(4)         -- TRUE.
1238     }
1239
1240
1241
1242
1243
1244 JmMediumTypeTC ::= TEXTUAL-CONVENTION
1245 STATUS current
1246 DESCRIPTION
1247     "Identifies the type of medium.
1248
1249     other(1),
1250         The type is neither one of the values listed in this specification nor a registered value.
1251
1252     unknown(2),
1253         The type is not known.
1254
1255     stationery(3),
1256         Separately cut sheets of an opaque material.
1257
1258     transparency(4),
1259         Separately cut sheets of a transparent material.
1260
1261     envelope(5),
1262         Envelopes that can be used for conventional mailing purposes.
1263
1264     envelopePlain(6),
1265         Envelopes that are not preprinted and have no windows.
1266
1267     envelopeWindow(7),
1268         Envelopes that have windows for addressing purposes.
1269
1270     continuousLong(8),
1271         Continuously connected sheets of an opaque material connected along the long edge.
1272
1273     continuousShort(9),
1274         Continuously connected sheets of an opaque material connected along the short edge.
1275

```

1276 **tabStock(10),**  
 1277 Media with tabs.  
 1278  
 1279 **multiPartForm(11),**  
 1280 Form medium composed of multiple layers not pre-attached to one another; each sheet  
 1281 MAY be drawn separately from an input source.  
 1282  
 1283 **labels(12),**  
 1284 Label-stock.  
 1285  
 1286 **multiLayer(13)**  
 1287 Form medium composed of multiple layers which are pre-attached to one another, e.g. for  
 1288 use with impact printers."  
 1289 REFERENCE  
 1290 "This is a type 2 enumeration. See Section 3.6.1.2."  
 1291 SYNTAX INTEGER {  
 1292 other(1),  
 1293 unknown(2),  
 1294 stationery(3),  
 1295 transparency(4),  
 1296 envelope(5),  
 1297 envelopePlain(6),  
 1298 envelopeWindow(7),  
 1299 continuousLong(8),  
 1300 continuousShort(9),  
 1301 tabStock(10),  
 1302 multiPartForm(11),  
 1303 labels(12),  
 1304 multiLayer(13)  
 1305 }  
 1306  
 1307  
 1308  
 1309  
 1310  
 1311 **JmJobSubmissionTypeTC ::= TEXTUAL-CONVENTION**  
 1312 STATUS current  
 1313 DESCRIPTION  
 1314 "Identifies the format type of a job submission ID.  
 1315  
 1316 Each job submission ID is a fixed-length, 48-octet printable US-ASCII [US-ASCII] coded  
 1317 character string containing no control characters, consisting of the following fields:  
 1318  
 1319 octet 1 The format letter identifying the format.  
 1320 \_\_\_\_\_The US-ASCII characters '0-9', 'A-Z', and 'a-z'  
 1321 \_\_\_\_\_are assigned in order giving 62 possible  
 1322 \_\_\_\_\_formats.  
 1323 octets 2-40 A 39-character, US-ASCII trailing SPACE filled

1324 field specified by the format letter, if the  
 1325 data is less than 39 ASCII characters.  
 1326 octets 41-48 A sequential or random number to make the ID  
 1327 quasi-unique.  
 1328

1329 If the client does not supply a job submission ID in the job submission protocol, then the  
 1330 agentserver SHALL assign a job submission ID using any of the standard formats that are  
 1331 reserved for to the agent. Clients SHALL not use formats that are reserved for to agents and  
 1332 agents SHALL NOT use formats that are reserved for clients, in order to reduce conflicts in ID  
 1333 generation. See the description for which formats are reserved for clients or for agents.  
 1334

1335 Registration of additional formats may be done following the procedures described in Section  
 1336 3.6.3.  
 1337

1338 The format values defined at the time of completion of this specification are:  
 1339

Format	
Letter	Description
-----	-----
'0'	octets 2-40: last 39 bytes of the <b>jmJobOwner</b> object. octets 41-48: 8-decimal-digit sequential number. This format is reserved <u>for to agents.</u> <del>for use when</del> <del>the client does not supply a job submission ID.</del>
	<u>NOTE - Clients wishing to use a job submission ID that incorporates the job owner, SHALL use format '8', not format '0', in order to reduce the chances of one client assigning the same ID as the agent when receiving a job from another client that does not supply a job submission id.</u>
	<del>NOTE - other formats may be registered that are reserved to the agent for use when the client does not supply a job submission ID.</del>
'1'	octets 2-40: last 39 bytes of the <b>jobName</b> attribute. octets 41-48: 8-decimal-digit random number. <u>This format is reserved for clients.</u>
'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 <u>This format is reserved for clients.</u>
'3'	octets 2-40: last 39 bytes of the client URL [URI-spec].

1373	octets 41-48: 8-decimal-digit sequential number
1374	<u>This format is reserved for clients.</u>
1375	
1376	'4' octets 2-40: last 39 bytes of the URI [URI-spec]
1377	assigned by the server or device to the job when
1378	the job was submitted for processing.
1379	octets 41-48: 8-decimal-digit sequential number
1380	<u>This format is reserved for agents.</u>
1381	
1382	'5' octets 2-40: last 39 bytes of a user number, such
1383	as POSIX user number.
1384	octets 41-48: 8-decimal-digit sequential number
1385	<u>This format is reserved for clients.</u>
1386	
1387	'6' octets 2-40: last 39 bytes of the user account
1388	number.
1389	octets 41-48: 8-decimal-digit sequential number
1390	<u>This format is reserved for clients.</u>
1391	
1392	'7' octets 2-40: last 39 bytes of the DTMF incoming
1393	FAX routing number.
1394	octets 41-48: 8-decimal-digit sequential number
1395	<u>This format is reserved for clients.</u>
1396	
1397	'8' octets 2-40: last 39 bytes of the job owner name
1398	(that the agent returns in the <b>jmJobOwner</b> object).
1399	octets 41-48: 8-decimal-digit sequential number
1400	<u>This format is reserved for clients.</u>
1401	
1402	'9' octets 2-40: last 39 bytes of the host name with
1403	<u>trailing SPACES that submitted the job to this</u>
1404	<u>server/device using a protocol, such as LPD</u>
1405	<u>[RFC-1179] which includes the host name in the job</u>
1406	<u>submission protocol.</u>
1407	<u>octets 41-48: 8-decimal-digit leading zero</u>
1408	<u>representation of the job id generated by the</u>
1409	<u>by the submitting server (configuration 3)</u>
1410	<u>or the client (configuration 1 and 2), such as in</u>
1411	<u>the LPD protocol.</u>
1412	<u>This format is reserved for clients.</u>
1413	

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



1422 intended to be an absolute guarantee of uniqueness. None-the-less, collisions are remotely  
 1423 possible, but without bad consequences, since this MIB is intended to be used only for  
 1424 monitoring jobs, not for controlling and managing them."

1425 REFERENCE

1426 "This is like a type 2 enumeration. See section 3.6.3."

1427 SYNTAX OCTET STRING(SIZE(1)) -- ASCII '0'-'9', 'A'-'Z', 'a'-'z'

1428  
 1429  
 1430  
 1431

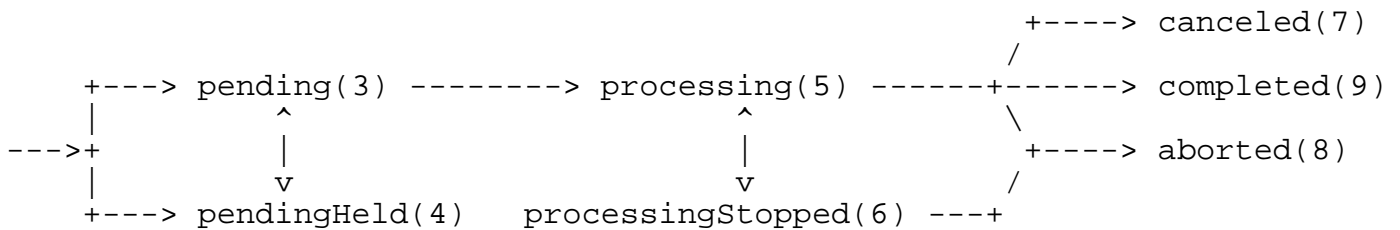
1432 **JmJobStateTC** ::= TEXTUAL-CONVENTION

1433 STATUS current

1434 DESCRIPTION

1435 "The current state of the job (**pending**, **processing**, **completed**, etc.).

1436  
 1437 The following figure shows the normal job state transitions:



1448 **Figure 4 - Normal Job State Transitions**

1449 Normally a job progresses from left to right. Other state transitions are unlikely, but are not  
 1450 forbidden. Not shown are the transitions to the **canceled** state from the **pending**,  
 1451 **pendingHeld**, **processing**, and **processingStopped** states.

1452  
 1453 Jobs in the **pending**, **processing**, and **processingStopped** states are called 'active', while jobs in  
 1454 the **pendingHeld**, **canceled**, **aborted**, and **completed** states are called 'inactive'. Jobs reach one  
 1455 of the three terminal states: **completed**, **canceled**, or **aborted**, after the jobs have completed all  
 1456 activity, and all MIB objects and attributes have reached their final values for the job.

1457  
 1458 These values are the same as the enum values of the IPP 'job-state' job attribute. See Section  
 1459 3.6.1.2.

1460 **unknown(2),**

1461 The job state is *not* known, or its state is indeterminate.

1462 **pending(3),**

1463 The job is a candidate to start processing, but is not yet processing.

1464  
 1465  
 1466  
 1467

1468  
1469  
1470  
1471  
1472  
1473  
1474  
1475  
1476  
1477  
1478  
1479  
1480  
1481  
1482  
1483  
1484  
1485  
1486  
1487  
1488  
1489  
1490  
1491  
1492  
1493  
1494  
1495  
1496  
1497  
1498  
1499  
1500  
1501  
1502  
1503  
1504  
1505  
1506  
1507  
1508  
1509  
1510  
1511  
1512  
1513  
1514  
1515

**pendingHeld(4),**

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

**processing(5),**

One of Either:

1. the job is using, or is attempting to use, one or more ~~document transforms which include (1)~~ purely software processes that are analyzing, creating, or interpreting a PDL, etc., and (2)

2. the job is using, or is attempting to use, one or more hardware devices that are interpreting a PDL, making marks on a medium, and/or performing finishing, such as stapling, etc.,

OR

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

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

Implementations MAY, though they NEED NOT, include additional values in the job's **jmJobStateReasons1** object to indicate the progress of the job, such as adding the **jobPrinting** value to indicate when the device is actually making marks on a medium and/or the **processingToStopPoint** value to indicate that the server or device is in the process of canceling or aborting the job.

**processingStopped(6),**

The job has stopped while processing for any number of reasons and will return to the **processing** state as soon as the reasons are no longer present.

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

NOTE - When an output device is stopped, the device usually indicates its condition in human readable form at the device. The management application can obtain more

1516 complete device status remotely by querying the appropriate device MIB using the job's  
 1517 **deviceIndex** attribute(s), if the agent implements such a device MIB  
 1518

1519 **canceled(7),**  
 1520 A client has canceled the job and ~~the job is either: (1) in the process of being terminated by~~  
 1521 ~~the server or device or (2) has completed canceling the job~~ and all MIB objects and  
 1522 attributes have reached their final values for the job ~~terminating~~. While the server or device  
 1523 is canceling the job, the job's **jmJobStateReasons1** object SHOULD contain the  
 1524 **processingToStopPoint** value and one of either the **canceledByUser**, or  
 1525 **canceledByOperator**, or **canceledAtDevice** values. The **canceledByUser**,  
 1526 **canceledByOperator**, or **canceledAtDevice** values remain while the job is in the  
 1527 **canceled** state.  
 1528

1529 **aborted(8),**  
 1530 The job has been aborted by the system, usually while the job was in the **processing** or  
 1531 **processingStopped** state and the server or device has completed aborting the job and all  
 1532 MIB objects and attributes have reached their final values for the job. While the server or  
 1533 device is aborting the job, the job's **jmJobStateReasons1** object MAY contain the  
 1534 **processingToStopPoint** and **abortedBySystem** values. If implemented, the  
 1535 **abortedBySystem** value SHALL remain while the job is in the **aborted** state.  
 1536

1537 **completed(9)**  
 1538 The job has completed successfully or with warnings or errors after processing and all of  
 1539 the media have been successfully stacked in the appropriate output bin(s). The job's  
 1540 **jmJobStateReasons1** object SHOULD contain one of: **completedSuccessfully**,  
 1541 **completedWithWarnings**, or **completedWithErrors** values."  
 1542 REFERENCE  
 1543 "This is a type 2 enumeration. See Section 3.6.1.2."  
 1544 SYNTAX INTEGER {  
 1545 unknown(2),  
 1546 pending(3),  
 1547 pendingHeld(4),  
 1548 processing(5),  
 1549 processingStopped(6),  
 1550 canceled(7),  
 1551 aborted(8),  
 1552 completed(9)  
 1553 }  
 1554

1555

1556 **JmAttributeTypeTC ::= TEXTUAL-CONVENTION**  
 1557 STATUS current  
 1558 DESCRIPTION  
 1559 "The type of the attribute which identifies the attribute."  
 1560  
 1561 In the following definitions of the enums, each description indicates whether the useful value of  
 1562 the attribute SHALL be represented using the **jmAttributeValueAsInteger** or the



1611 **jmJobState** object. See the description under **JmJobStateReasons1TC** textual-  
1612 convention.

1613  
1614 **jobStateReasons4(5),** **JmJobStateReasons4TC**  
1615 INTEGER: Additional information about the job's current state that augments the  
1616 **jmJobState** object. See the description under **JmJobStateReasons1TC** textual-  
1617 convention.

1618  
1619 **processingMessage(6),** **JmUTF8StringTC(SIZE(0..63))**  
1620 OCTETS: MULTI-ROW: A coded character set message that is generated by the server  
1621 or device during the processing of the job as a simple form of processing log to show  
1622 progress and any problems.

1623  
1624 There is no restriction for the same message occurring in multiple rows.

1625  
1626 **jobCodedCharSet(7),** **CodedCharSet**  
1627 INTEGER: The MIBenum identifier of the coded character set that the agent is using to  
1628 represent coded character set objects and attributes of type '**JmJobStringTC**'. These  
1629 coded character set objects and attributes are either: (1) supplied by the job submitting  
1630 client or (2) defaulted by the server or device when omitted by the job submitting client.  
1631 The agent SHALL represent these objects and attributes in the MIB either (1) in the coded  
1632 character set as they were submitted or (2) MAY convert the coded character set to  
1633 another coded character set or encoding scheme as identified by the **jobCodedCharSet**  
1634 attribute.

1635  
1636 These MIBenum values are assigned by IANA [IANA-charsets] when the coded character  
1637 sets are registered. The coded character set SHALL be one of the ones registered with  
1638 IANA [IANA] and the enum value uses the **CodedCharSet** textual-convention from the  
1639 Printer MIB. See the **JmJobStringTC** textual-convention.

1640  
1641 If the agent does not know what coded character set was used by the job submitting client,  
1642 the agent SHALL either (1) return the '**unknown(2)**' value for the **jobCodedCharSet**  
1643 attribute or (2) not return the **jobCodedCharSet** attribute for the job. See Section 3.5.2,  
1644 entitled 'Text generated by the job submitter'.

1645  
1646  
1647  
1648 ++++++  
1649 + **Job Identification attributes**  
1650 +  
1651 + **The following attributes help an end user, a system**  
1652 + **operator, or an accounting program identify a job.**  
1653 ++++++

1654  
1655  
1656  
1657 **jobURI(20),** **OCTET STRING(SIZE(1..255))**  
1658 OCTETS: The job's Universal Resource Identifier (URI) [RFC-1738]. See IPP for  
1659 example usage.

1660  
1661  
1662  
1663  
1664  
1665  
1666  
1667  
1668  
1669  
1670  
1671  
1672  
1673  
1674  
1675  
1676  
1677  
1678  
1679  
1680  
1681  
1682  
1683  
1684  
1685  
1686  
1687  
1688  
1689  
1690  
1691  
1692  
1693  
1694  
1695  
1696  
1697  
1698  
1699  
1700  
1701  
1702  
1703  
1704  
1705  
1706  
1707  
1708

NOTE - The agent may be able to generate this value on each SNMP Get operation from smaller values, rather than having to store the entire URI.

If the URI exceeds 255 octets, the agent SHALL truncate from the beginning (since the end tends to be more unique than the beginning).

**jobAccountName(21),** **OCTET**

**STRINGJmJobStringTC(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),** **JmJobStringTC(SIZE(0..63))**

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

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

**jobName(23),** **JmJobStringTC(SIZE(0..63))**

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

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

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

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

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

1709		
1710	<b>jobServiceTypes(24),</b>	<b>JmJobServiceTypesTC</b>
1711	INTEGER: Specifies the type(s) of service to which the job has been submitted (print, fax, scan, etc.). The service type is bit encoded with each job service type so that more general and arbitrary services can be created, such as services with more than one destination type, or ones with only a source or only a destination. For example, a job service might <b>scan</b> , <b>faxOut</b> , and <b>print</b> a single job. In this case, three bits would be set in the <b>jobServiceTypes</b> attribute, corresponding to the hexadecimal values: <b>0x8</b> + <b>0x20</b> + <b>0x4</b> , respectively, yielding: <b>0x2C</b> .	
1712		
1713		
1714		
1715		
1716		
1717		
1718		
1719	Whether this attribute is set from a job attribute supplied by the job submission client or is set by the recipient job submission server or device depends on the job submission protocol. This attribute SHALL be implemented if the server or device has other types in addition to or instead of printing.	
1720		
1721		
1722		
1723		
1724	One of the purposes of this attribute is to permit a requester to filter out jobs that are not of interest. For example, a printer operator may only be interested in jobs that include printing.	
1725		
1726		
1727		
1728	<b>jobSourceChannelIndex(25),</b>	Integer32(0..2147483647)
1729	INTEGER: The index of the row in the associated Printer MIB[print-mib] of the channel which is the source of the print job.	
1730		
1731		
1732	<b>jobSourcePlatformType(26),</b>	<b>JmJobSourcePlatformTypeTC</b>
1733	INTEGER: The source platform type of the immediate upstream submitter that submitted the job to the server (configuration 2) or device (configuration 1 and 3) to which the agent is providing access. For configuration 1, this is the type of the client that submitted the job to the device; for configuration 2, this is the type of the client that submitted the job to the server; and for configuration 3, this is the type of the server that submitted the job to the device.	
1734		
1735		
1736		
1737		
1738		
1739		
1740	<b>submittingServerName(27),</b>	<b>JmJobStringTC(SIZE(0..63))</b>
1741	OCTETS: For configuration 3 only: The administrative name of the server that submitted the job to the device.	
1742		
1743		
1744	<b>submittingApplicationName(28),</b>	<b>JmJobStringTC(SIZE(0..63))</b>
1745	OCTETS: The name of the client application (not the server in configuration 3) that submitted the job to the server or device.	
1746		
1747		
1748	<b>jobOriginatingHost(29),</b>	<b>JmJobStringTC(SIZE(0..63))</b>
1749	OCTETS: The name of the client host (not the server host name in configuration 3) that submitted the job to the server or device.	
1750		
1751		
1752	<b>deviceNameRequested(30),</b>	<b>JmJobStringTC(SIZE(0..63))</b>
1753	OCTETS: The administratively defined coded character set name of the target device requested by the submitting user. For configuration 1, its value corresponds to the Printer MIB[print-mib]: <b>prtGeneralPrinterName</b> object. For configuration 2 and 3, its value is the name of the logical or physical device that the user supplied to indicate to the server on which device(s) they wanted the job to be processed.	
1754		
1755		
1756		
1757		

1758  
1759 **queueNameRequested(31),** **JmJobStringTC(SIZE(0..63))**  
1760 OCTETS: The administratively defined coded character set name of the target queue  
1761 requested by the submitting user. For configuration 1, its value corresponds to the queue  
1762 in the device for which the agent is providing access. For configuration 2 and 3, its value  
1763 is the name of the queue that the user supplied to indicate to the server on which device(s)  
1764 they wanted the job to be processed.  
1765  
1766 NOTE - typically an implementation SHOULD support either the **deviceNameRequested**  
1767 or **queueNameRequested** attribute, but not both.  
1768  
1769 **physicalDevice(32),** **hrDeviceIndex**  
1770 AND/OR  
1771 **JmUTF8StringTC(SIZE(0..63))**  
1772 INTEGER: MULTI-ROW: The index of the physical device MIB instance  
1773 requested/used, such as the Printer MIB[print-mib]. This value is an **hrDeviceIndex**  
1774 value. See the Host Resources MIB[hr-mib].  
1775  
1776 AND/OR  
1777  
1778 OCTETS: MULTI-ROW: The name of the physical device to which the job is assigned.  
1779  
1780 **numberOfDocuments(33),** **Integer32(-2..2147483647)**  
1781 INTEGER: The number of documents in this job.  
1782  
1783 **fileName(34),** **JmJobStringTC(SIZE(0..63))**  
1784 OCTETS: MULTI-ROW: The coded character set file name or URI[URI-spec] of the  
1785 document.  
1786  
1787 There is no restriction on the same file name occurring in multiple rows.  
1788  
1789 **documentName(35),** **JmJobStringTC(SIZE(0..63))**  
1790 OCTETS: MULTI-ROW: The coded character set name of the document.  
1791  
1792 There is no restriction on the same document name occurring in multiple rows.  
1793  
1794 **jobComment(36),** **JmJobStringTC(SIZE(0..63))**  
1795 OCTETS: An arbitrary human-readable coded character text string supplied by the  
1796 submitting user or the job submitting application program for any purpose. For example,  
1797 a user might indicate what he/she is going to do with the printed output or the job  
1798 submitting application program might indicate how the document was produced.  
1799  
1800 The **jobComment** attribute is not intended to be a name; see the **jobName** attribute.  
1801  
1802 **documentFormatIndex(37),** **Integer32(0..2147483647)**  
1803 INTEGER: MULTI-ROW: The index in the **prtInterpreterTable** in the Printer  
1804 MIB[print-mib] of the page description language (PDL) or control language interpreter  
1805 that this job requires/uses. A document or a job MAY use more than one PDL or control  
1806 language.



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  
1833  
1834  
1835  
1836  
1837  
1838  
1839  
1840  
1841  
1842  
1843  
1844  
1845  
1846  
1847  
1848  
1849  
1850  
1851  
1852  
1853  
1854  
1855

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

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

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

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

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**.

1856 **jobHold(52),** **JmBooleanTC**  
 1857 INTEGER: If the value is 'true(4)', a client has explicitly specified that the job is to be  
 1858 held until explicitly released. Until the job is explicitly released by a client, the job SHALL  
 1859 be in the **pendingHeld** state with the **jobHoldSpecified** value in the  
 1860 **jmJobStateReasons1** attribute.  
 1861

1862 **jobHoldUntil(53),** **JmJobStringTC(SIZE(0..63))**  
 1863 OCTETS: The named time period during which the job SHALL become a candidate for  
 1864 processing, such as 'evening', 'night', 'weekend', 'second-shift', 'third-shift', etc., as  
 1865 defined by the system administrator. See IPP [ipp-model] for the standard keyword  
 1866 values. Until that time period arrives, the job SHALL be in the **pendingHeld** state with  
 1867 the **jobHoldUntilSpecified** value in the **jmJobStateReasons1** object. The value 'no-  
 1868 hold' SHALL indicate explicitly that no time period has been specified; the absence of this  
 1869 attribute SHALL indicate implicitly that no time period has been specified.  
 1870

1871 **outputBin(54),** **Integer32(0..2147483647)**  
 1872 **AND/OR**  
 1873 **JmJobStringTC(SIZE(0..63))**  
 1874 INTEGER: MULTI-ROW: The output subunit index in the Printer MIB[print-mib]  
 1875  
 1876 **AND/OR**  
 1877  
 1878 OCTETS: the name or number (represented as ASCII digits) of the output bin to which  
 1879 all or part of the job is placed in.  
 1880

1881 **sides(55),** **Integer32(-2..2)**  
 1882 INTEGER: MULTI-ROW: The number of sides, '1' or '2', that any document in this job  
 1883 requires/used.  
 1884

1885 **finishing(56),** **JmFinishingTC**  
 1886 INTEGER: MULTI-ROW: Type of finishing that any document in this job requires/used.  
 1887  
 1888  
 1889 ++++++  
 1890 + **Image Quality attributes (requested and consumed)**  
 1891 +  
 1892 + **For devices that can vary the image quality.**  
 1893 ++++++

1894

1895 **printQualityRequested(70),** **JmPrintQualityTC**  
 1896 INTEGER: MULTI-ROW: The print quality selection requested for a document in the  
 1897 job for printers that allow quality differentiation.  
 1898

1899 **printQualityUsed(71),** **JmPrintQualityTC**  
 1900 INTEGER: MULTI-ROW: The print quality selection actually used by a document in the  
 1901 job for printers that allow quality differentiation.  
 1902

1903 **printerResolutionRequested(72),** **JmPrinterResolutionTC**  
1904 OCTETS: MULTI-ROW: The printer resolution requested for a document in the job for  
1905 printers that support resolution selection.  
1906

1907 **printerResolutionUsed(73),** **JmPrinterResolutionTC**  
1908 OCTETS: MULTI-ROW: The printer resolution actually used by a document in the job  
1909 for printers that support resolution selection.  
1910

1911 **tonerEcomonyRequested(74),** **JmTonerEcomonyTC**  
1912 INTEGER: MULTI-ROW: The toner economy selection requested for documents in the  
1913 job for printers that allow toner economy differentiation.  
1914

1915 **tonerEcomonyUsed(75),** **JmTonerEcomonyTC**  
1916 INTEGER: MULTI-ROW: The toner economy selection actually used by documents in  
1917 the job for printers that allow toner economy differentiation.  
1918

1919 **tonerDensityRequested(76),** **Integer32(-2..100)**  
1920 INTEGER: MULTI-ROW: The toner density requested for a document in this job for  
1921 devices that can vary toner density levels. Level 1 is the lowest density and level 100 is  
1922 the highest density level. Devices with a smaller range, SHALL map the 1-100 range  
1923 evenly onto the implemented range.  
1924

1925 **tonerDensityUsed(77),** **Integer32(-2..100)**  
1926 INTEGER: MULTI-ROW: The toner density used by documents in this job for devices  
1927 that can vary toner density levels. Level 1 is the lowest density and level 100 is the highest  
1928 density level. Devices with a smaller range, SHALL map the 1-100 range evenly onto the  
1929 implemented range.  
1930

1931  
1932 ++++++  
1933 + **Job Progress attributes (requested and consumed)**  
1934 +  
1935 + **Pairs of these attributes can be used by monitoring**  
1936 + **applications to show an indication of relative progress**  
1937 + **to users.**  
1938 ++++++

1939  
1940 **jobCopiesRequested(90),** **Integer32(-2..2147483647)**  
1941 INTEGER: The number of copies of the entire job that are to be produced.  
1942

1943 **jobCopiesCompleted(91),** **Integer32(-2..2147483647)**  
1944 INTEGER: The number of copies of the entire job that have been completed so far.  
1945

1946 **documentCopiesRequested(92),** **Integer32(-2..2147483647)**  
1947 INTEGER: The total count of the number of document copies requested for the job as a  
1948 whole. If there are documents A, B, and C, and document B is specified to produce 4  
1949 copies, the number of document copies requested is 6 for the job.  
1950

1951 This attribute SHALL be used only when a job has multiple documents. The  
1952 **jobCopiesRequested** attribute SHALL be used when the job has only one document.  
1953

1954 **documentCopiesCompleted(93), Integer32(-2..2147483647)**

1955 INTEGER: The total count of the number of document copies completed so far for the  
1956 job as a whole. If there are documents A, B, and C, and document B is specified to  
1957 produce 4 copies, the number of document copies starts a 0 and runs up to 6 for the job as  
1958 the job processes.  
1959

1960 This attribute SHALL be used only when a job has multiple documents. The  
1961 **jobCopiesCompleted** attribute SHALL be used when the job has only one document.  
1962

1963 **jobKOctetsTransferred(94), Integer32(-2..2147483647)**

1964 INTEGER: The number of K (1024) octets transferred to the server or device to which  
1965 the agent is providing access. This count is independent of the number of copies of the  
1966 job or documents that will be produced, but it is only a measure of the number of bytes  
1967 transferred to the server or device.  
1968

1969 The agent SHALL round the actual number of octets transferred up to the next higher K.  
1970 Thus 0 octets SHALL be represented as '0', 1-1024 octets SHALL BE represented as '1',  
1971 1025-2048 SHALL be '2', etc. When the job completes, the values of the  
1972 **jmJobKOctetsRequested** object and the **jobKOctetsTransferred** attribute SHALL be  
1973 equal.  
1974

1975 NOTE - The **jobKOctetsTransferred** can be used with the **jmJobKOctetsRequested**  
1976 object in order to produce a relative indication of the progress of the job for agents that do  
1977 not implement the **jmJobKOctetsProcessed** object.  
1978  
1979

1980 ++++++  
1981 + **Impression attributes**  
1982 +  
1983 + **For a print job, an impression is the marking of the**  
1984 + **entire side of a sheet. Two-sided processing involves two**  
1985 + **impressions per sheet. Two-up is the placement of two**  
1986 + **logical pages on one side of a sheet and so is still a**  
1987 + **single impression. See also jmJobImpressionsRequested and**  
1988 + **jmJobImpressionsCompleted objects in the jmJobTable.**  
1989 ++++++

1990  
1991 **impressionsSpooled(110), Integer32(-2..2147483647)**

1992 INTEGER: The number of impressions spooled to the server or device for the job so far.  
1993

1994 **impressionsSentToDevice(111), Integer32(-2..2147483647)**

1995 INTEGER: The number of impressions sent to the device for the job so far.  
1996

1997 **impressionsInterpreted(112), Integer32(-2..2147483647)**

1998 INTEGER: The number of impressions interpreted for the job so far.  
1999

2000  
2001  
2002  
2003  
2004  
2005  
2006  
2007  
2008  
2009  
2010  
2011  
2012  
2013  
2014  
2015  
2016  
2017  
2018  
2019  
2020  
2021  
2022  
2023  
2024  
2025  
2026  
2027  
2028  
2029  
2030  
2031  
2032  
2033  
2034  
2035  
2036  
2037  
2038  
2039  
2040  
2041  
2042  
2043  
2044  
2045  
2046

**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.

**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.

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

2047 NOTE - See the **impressionsCompletedCurrentCopy** and  
2048 **pagesCompletedCurrentCopy** attributes for attributes that are reset on each document  
2049 copy.  
2050

2051 NOTE - The **pagesCompleted** object can be used with the **pagesRequested** object to  
2052 provide an indication of the relative progress of the job, provided that the multiplicative  
2053 factor is taken into account for some implementations of multiple copies.  
2054

2055 **pagesCompletedCurrentCopy(132), Integer32(-2..2147483647)**  
2056 INTEGER: The number of logical pages completed for the current copy of the document  
2057 so far. This value SHALL be reset to **0** for each document in the job and for each  
2058 document copy.  
2059

2061 ++++++  
2062 + **Sheet attributes**  
2063 +  
2064 + **The sheet is a single piece of a medium, whether printing**  
2065 + **on one or both sides.**  
2066 ++++++

2067  
2068 **sheetsRequested(150), Integer32(-2..2147483647)**  
2069 INTEGER: The number of medium sheets requested to be processed for this job.  
2070

2071 **sheetsCompleted(151), Integer32(-2..2147483647)**  
2072 INTEGER: The number of medium sheets that have completed marking and stacking for  
2073 the entire job so far whether those sheets have been processed on one side or on both.  
2074

2075 **sheetsCompletedCurrentCopy(152), Integer32(-2..2147483647)**  
2076 INTEGER: The number of medium sheets that have completed marking and stacking for  
2077 the current copy of a document in the job so far whether those sheets have been processed  
2078 on one side or on both.  
2079

2080 The value of this attribute SHALL be reset to **0** as each document in the job starts being  
2081 processed and for each document copy as it starts being processed.  
2082

2083  
2084 ++++++  
2085 + **Resources attributes (requested and consumed)**  
2086 +  
2087 + **Pairs of these attributes can be used by monitoring**  
2088 + **applications to show an indication of relative usage to**  
2089 + **users.**  
2090 ++++++

2091 **mediumRequested(170), JmMediumTypeTC**  
2092 **AND/OR**  
2093 **JmJobStringTC(SIZE(0..63))**  
2094  
2095 INTEGER: MULTI-ROW: The type

2096 AND/OR  
 2097 OCTETS: the name of the medium that is required by the job.  
 2098  
 2099 **mediumConsumed(171), Integer32(-2..2147483647)**  
 2100 AND  
 2101 **JmJobStringTC(SIZE(0..63))**  
 2102 INTEGER: The number of sheets  
 2103 AND  
 2104 OCTETS: MULTI-ROW: the name of the medium that has been consumed so far  
 2105 whether those sheets have been processed on one side or on both.  
 2106  
 2107 This attribute SHALL have both **Integer32** and **OCTET STRING** (represented as  
 2108 **JmJobStringTC**) values.  
 2109  
 2110 **colorantRequested(172), Integer32(-2..2147483647)**  
 2111 AND/OR  
 2112 **JmJobStringTC(SIZE(0..63))**  
 2113 INTEGER: MULTI-ROW: The index (**prtMarkerColorantIndex**) in the Printer  
 2114 MIB[print-mib]  
 2115 AND/OR  
 2116 OCTETS: the name of the colorant requested.  
 2117  
 2118 **colorantConsumed(173), Integer32(-2..2147483647)**  
 2119 AND/OR  
 2120 **JmJobStringTC(SIZE(0..63))**  
 2121 INTEGER: MULTI-ROW: The index (**prtMarkerColorantIndex**) in the Printer  
 2122 MIB[print-mib]  
 2123 AND/OR  
 2124 OCTETS: the name of the colorant consumed.  
 2125  
 2126  
 2127 ++++++  
 2128 + **Time attributes (set by server or device)**  
 2129 +  
 2130 + **This section of attributes are ones that are set by the**  
 2131 + **server or device that accepts jobs. Two forms of time are**  
 2132 + **provided. Each form is represented in a separate attribute.**  
 2133 + **See section 3.1.2 and section 3.1.3 for the**  
 2134 + **conformance requirements for time attribute for agents and**  
 2135 + **monitoring applications, respectively. The two forms are:**  
 2136 +  
 2137 + **'DateAndTime' is an 8 or 11 octet binary encoded year,**  
 2138 + **month, day, hour, minute, second, deci-second with**  
 2139 + **optional offset from UTC. See SNMPv2-TC [SMIv2-TC].**  
 2140 +  
 2141 + **NOTE: 'DateAndTime' is not printable characters; it is**  
 2142 + **binary.**  
 2143 +  
 2144 + **'JmTimeStampTC' is the time of day measured in the number of**

2145 + seconds since the system was booted.  
 2146 ++++++

2147

2148 **jobSubmissionToServerTime(190),** **JmTimeStampTC**  
 2149 AND/OR  
 2150 **DateAndTime**  
 2151 INTEGER: Configuration 3 only: The time  
 2152 AND/OR  
 2153 OCTETS: the date and time that the job was submitted to the server (as distinguished  
 2154 from the device which uses jobSubmissionTime).  
 2155

2156 **jobSubmissionTime(191),** **JmTimeStampTC**  
 2157 AND/OR  
 2158 **DateAndTime**  
 2159 INTEGER: Configurations 1, 2, and 3: The time  
 2160 AND/OR  
 2161 OCTETS: the date and time that the job was submitted to the server or device to which  
 2162 the agent is providing access.  
 2163  
 2164  
 2165

2166 **jobStartedBeingHeldTime(192),** **JmTimeStampTC**  
 2167 AND/OR  
 2168 **DateAndTime**  
 2169 INTEGER: The time  
 2170 AND/OR  
 2171 OCTETS: the date and time that the job last entered the **pendingHeld** state. If the job  
 2172 has never entered the **pendingHeld** state, then the value SHALL be '0' or the attribute  
 2173 SHALL not be present in the table.  
 2174

2175 **jobStartedProcessingTime(193),** **JmTimeStampTC**  
 2176 AND/OR  
 2177 **DateAndTime**  
 2178 INTEGER: The time  
 2179 AND/OR  
 2180 OCTETS: the date and time that the job started processing.  
 2181

2182 **jobCompletionedTime(194),** **JmTimeStampTC**  
 2183 AND/OR  
 2184 **DateAndTime**  
 2185 INTEGER: The time  
 2186 AND/OR  
 2187 OCTETS: the date and time that the job entered the **completed, canceled, or aborted**  
 2188 state.  
 2189

2190 **jobProcessingCPUTime(195)** **Integer32(-2..2147483647)**  
 2191 **UNITS 'seconds'**  
 2192 INTEGER: The amount of CPU time in seconds that the job has been in the **processing**  
 2193 state. If the job enters the **processingStopped** state, that elapsed time SHALL not be



2194 included. In other words, the **jobProcessingCPUTime** value SHOULD be relatively  
 2195 repeatable when the same job is processed again on the same device."  
 2196

2197 REFERENCE

2198 "See Section 3.2 entitled 'The Attribute Mechanism' for a description of this textual-convention  
 2199 and its use in the **jmAttributeTable**.

2200 This is a type 2 enumeration. See Section 3.6.1.2."

2201 SYNTAX INTEGER {

2202 other(1),  
 2203 unknown(2),  
 2204 jobStateReasons2(3),  
 2205 jobStateReasons3(4),  
 2206 jobStateReasons4(5),  
 2207 processingMessage(6),  
 2208 jobCodedCharSet(7),  
 2209  
 2210 jobURI(20),  
 2211 jobAccountName(21),  
 2212 serverAssignedJobName(22),  
 2213 jobName(23),  
 2214 jobServiceTypes(24),  
 2215 jobSourceChannelIndex(25),  
 2216 jobSourcePlatformType(26),  
 2217 submittingServerName(27),  
 2218 submittingApplicationName(28),  
 2219 jobOriginatingHost(29),  
 2220 deviceNameRequested(30),  
 2221 queueNameRequested(31),  
 2222 physicalDevice(32),  
 2223 numberOfDocuments(33),  
 2224 fileName(34),  
 2225 documentName(35),  
 2226 jobComment(36),  
 2227 documentFormatIndex(37),  
 2228 documentFormat(38),  
 2229  
 2230  
 2231 jobPriority(50),  
 2232 jobProcessAfterDateAndTime(51),  
 2233 jobHold(52),  
 2234 jobHoldUntil(53),  
 2235 outputBin(54),  
 2236 sides(55),  
 2237 finishing(56),  
 2238  
 2239 printQualityRequested(70),  
 2240 printQualityUsed(71),  
 2241 printerResolutionRequested(72),  
 2242 printerResolutionUsed(73),

```

2243         tonerEcomonyRequested(74),
2244         tonerEcomonyUsed(75),
2245         tonerDensityRequested(76),
2246         tonerDensityUsed(77),
2247
2248         jobCopiesRequested(90),
2249         jobCopiesCompleted(91),
2250         documentCopiesRequested(92),
2251         documentCopiesCompleted(93),
2252         jobKOctetsTransferred(94),
2253
2254         impressionsSpooled(110),
2255         impressionsSentToDevice(111),
2256         impressionsInterpreted(112),
2257         impressionsCompletedCurrentCopy(113),
2258         fullColorImpressionsCompleted(114),
2259         highlightColorImpressionsCompleted(115),
2260
2261         pagesRequested(130),
2262         pagesCompleted(131),
2263         pagesCompletedCurrentCopy(132),
2264
2265         sheetsRequested(150),
2266         sheetsCompleted(151),
2267         sheetsCompletedCurrentCopy(152),
2268
2269         mediumRequested(170),
2270         mediumConsumed(171),
2271         colorantRequested(172),
2272         colorantConsumed(173),
2273
2274         jobSubmissionToServerTime(190),
2275         jobSubmissionTime(191),
2276         jobStartedBeingHeldTime(192),
2277         jobStartedProcessingTime(193),
2278         jobCompletionTime(194),
2279         jobProcessingCPUTime(195)
2280     }

```

2285 **JmJobServiceTypesTC ::= TEXTUAL-CONVENTION**

2286     STATUS     current

2287     DESCRIPTION

2288         "Specifies the type(s) of service to which the job has been submitted (print, fax, scan, etc.). The  
2289         service type is represented as an enum that is bit encoded with each job service type so that  
2290         more general and arbitrary services can be created, such as services with more than one

2291 destination type, or ones with only a source or only a destination. For example, a job service  
 2292 might **scan**, **faxOut**, and **print** a single job. In this case, three bits would be set in the  
 2293 **jobServiceTypes** attribute, corresponding to the hexadecimal values: **0x8** + **0x20** + **0x4**,  
 2294 respectively, yielding: **0x2C**.  
 2295

2296 Whether this attribute is set from a job attribute supplied by the job submission client or is set by  
 2297 the recipient job submission server or device depends on the job submission protocol. With  
 2298 either implementation, the agent SHALL return a non-zero value for this attribute indicating the  
 2299 type of the job.  
 2300

2301 One of the purposes of this attribute is to permit a requester to filter out jobs that are not of  
 2302 interest. For example, a printer operator MAY only be interested in jobs that include printing.  
 2303 That is why the attribute is in the job identification category.  
 2304

2305 The following service component types are defined (in hexadecimal) and are assigned a separate  
 2306 bit value for use with the **jobServiceTypes** attribute:  
 2307

2308 **other 0x1**

2309 The job contains some instructions that are not one of the identified types.  
 2310

2311 **unknown 0x2**

2312 The job contains some instructions whose type is unknown to the agent.  
 2313

2314 **print 0x4**

2315 The job contains some instructions that specify printing  
 2316

2317 **scan 0x8**

2318 The job contains some instructions that specify scanning  
 2319

2320 **faxIn 0x10**

2321 The job contains some instructions that specify receive fax  
 2322

2323 **faxOut 0x20**

2324 The job contains some instructions that specify sending fax  
 2325

2326 **getFile 0x40**

2327 The job contains some instructions that specify accessing files or documents  
 2328

2329 **putFile 0x80**

2330 The job contains some instructions that specify storing files or documents  
 2331

2332 **mailList 0x100**

2333 The job contains some instructions that specify distribution of documents using an  
 2334 electronic mail system."  
 2335

2336 REFERENCE

2337 "These bit definitions are the equivalent of a type 2 enum except that combinations of them  
 2338 MAY be used together. See section 3.6.1.2."

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

2339

2340

2341

2342

2343 **JmJobStateReasons1TC** ::= TEXTUAL-CONVENTION

2344 STATUS current

2345 DESCRIPTION

2346 "The **JmJobStateReasonsN**TC ( $N=1..4$ ) textual-conventions are used with the  
 2347 **jmJobStateReasons1** object and **jobStateReasonsN** ( $N=2..4$ ), respectively, to provide  
 2348 additional information regarding the current **jmJobState** object value. These values MAY be  
 2349 used with any job state or states for which the reason makes sense.

2350

2351 NOTE - While values cannot be added to the **jmJobState** object without impacting deployed  
 2352 clients that take actions upon receiving **jmJobState** values, it is the intent that additional  
 2353 **JmJobStateReasonsN**TC enums can be defined and registered without impacting such  
 2354 deployed clients. In other words, the **jmJobStateReasons1** object and **jobStateReasonsN**  
 2355 attributes are intended to be extensible.

2356

2357 NOTE - The Job Monitoring MIB contains a superset of the IPP values[ipp-model] for the IPP  
 2358 'job-state-reasons' attribute, since the Job Monitoring MIB is intended to cover other job  
 2359 submission protocols as well. Also some of the names of the reasons have been changed from  
 2360 'printer' to 'device', since the Job Monitoring MIB is intended to cover additional types of  
 2361 devices, including input devices, such as scanners.

2362

2363 The following standard values are defined (in hexadecimal) as *powers of two*, since multiple  
 2364 values MAY be used at the same time. For ease of understanding, the  
 2365 **JmJobStateReasons1TC** reasons are presented in the order in which the reasons are likely to  
 2366 occur (if implemented), starting with the '**jobIncoming**' value and ending with the  
 2367 '**jobCompletedWithErrors**' value.

2368

2369 **other** **0x1**

The job state reason is not one of the standardized or registered reasons.

2370

2371 **unknown** **0x2**

The job state reason is not known to the agent or is indeterminate.

2372

2373 **jobIncoming** **0x4**

The job has been accepted by the server or device, but the server or device is expecting  
 (1) additional operations from the client to finish creating the job and/or (2) is  
 accessing/accepting document data.

2376

2377 **submissionInterrupted** **0x8**

The job was not completely submitted for some unforeseen reason, such as: (1) the server  
 has crashed before the job was closed by the client, (2) the server or the document transfer  
 method has crashed in some non-recoverable way before the document data was entirely

2378

2379

2380

2381

2382

2383

2384		<u>transferred to the server, (3) the client crashed or failed to close the job before the time-</u>
2385		<u>out period.</u>
2386		
2387	<b>jobOutgoing</b>	<b>0x108</b>
2388		Configuration 2 only: The server is transmitting the job to the device.
2389		
2390	<b>jobHoldSpecified</b>	<b>0x2010</b>
2391		The value of the job's jobHold(52) attribute is TRUE. The job SHALL NOT be a
2392		candidate for processing until this reason is removed and there are no other reasons to
2393		hold the job.
2394		
2395	<b>jobHoldUntilSpecified</b>	<b>0x4020</b>
2396		The value of the job's jobHoldUntil(53) attribute specifies a time period that is still in the
2397		future. The job SHALL NOT be a candidate for processing until this reason is removed
2398		and there are no other reasons to hold the job.
2399		
2400	<b>jobProcessAfterSpecified</b>	<b>0x8040</b>
2401		The value of the job's jobProcessAfterDateAndTime(51) attribute specifies a time that is
2402		still in the future. The job SHALL NOT be a candidate for processing until this reason is
2403		removed and there are no other reasons to hold the job.
2404		
2405	<b>resourcesAreNotReady</b>	<b>0x10080</b>
2406		At least one of the resources needed by the job, such as media, fonts, resource objects,
2407		etc., is not ready on any of the physical devices for which the job is a candidate. This
2408		condition MAY be detected when the job is accepted, or subsequently while the job is
2409		<b>pending</b> or <b>processing</b> , depending on implementation.
2410		
2411	<b>deviceStoppedPartly</b>	<b>0x200100</b>
2412		One or more, but not all, of the devices to which the job is assigned are stopped. If all of
2413		the devices are stopped (or the only device is stopped), the <b>deviceStopped</b> reason
2414		SHALL be used.
2415		
2416	<b>deviceStopped</b>	<b>0x400200</b>
2417		The device(s) to which the job is assigned is (are all) stopped.
2418		
2419	<b>jobInterpreting</b>	<b>0x800</b>
2420		<u>The device to which the job is assigned is interpreting the document data.</u>
2421		
2422	<b>jobPrinting</b>	<b>0x1000400</b>
2423		The output device <u>to which the job is assigned</u> is marking media. This attribute is useful
2424		for servers and output devices which spend a great deal of time processing (1) when no
2425		marking is happening and then want to show that marking is now happening or (2) when
2426		the job is in the <u>process of being canceled or aborted while the job remains in the</u>
2427		<b>processing</b> state, but the marking has not yet stopped so that impression or sheet counts
2428		are still increasing for the job.
2429		
2430	<b>jobCanceledByUser</b>	<b>0x2000800</b>
2431		The job was canceled by the <u>owner of the job</u> user, i.e., by <del>an unknown user or</del> by a user

2432	whose name is the same as the value of the job's <b>jmJobOwner</b> object, or by some other	
2433	<u>authorized end-user, such as a member of the job owner's security group.</u>	
2434		
2435	<b>jobCanceledByOperator</b>	<b>0x41000</b>
2436	The job was canceled by the operator, i.e., by a user <u>who has been authenticated as having</u>	
2437	<u>operator privileges (whether local or remote) whose name is different than the value of the</u>	
2438	<u>job's <b>jmJobOwner</b> object.</u>	
2439		
2440	<b>jobCanceledAtDevice</b>	<b>0x8000</b>
2441	<u>The job was canceled by an unidentified local user, i.e., a user at a console at the device.</u>	
2442		
2443	<b>abortedBySystem</b>	<b>0x100002000</b>
2444	The job (1) <u>is in the process of being aborted, (2) has been</u> was aborted by the system <u>and</u>	
2445	<u>placed in the '<b>aborted</b>' state, or (3) has been aborted by the system and placed -</u>	
2446		
2447	<del>NOTE—When the system puts a job into the '<b>aborted</b>' job state, this reason is not needed. This reason is</del>	
2448	<del>needed only when the system aborts a job, but, instead of placing the job in the <b>aborted</b> job state, places</del>	
2449	<del>the job in the '<b>pendingHeld</b>' state, so that a user or operator can manually try the job again.</del>	
2450		
2451	<b>processingToStopPoint</b>	<b>0x200004000</b>
2452	The requester has issued an operation to cancel or interrupt the job or the server/device	
2453	has aborted the job, but the server/device is still performing some actions on the job until a	
2454	specified stop point occurs or job termination/cleanup is completed.	
2455		
2456	This reason is recommended to be used in conjunction with the <b>processingcanceled</b> or	
2457	<b>aborted</b> job state to indicate that the server/device is still performing some actions on the	
2458	job <u>while</u> after the job <u>remains in</u> leaves the <b>processing</b> state. <u>After all the , so that some of</u>	
2459	<u>the job's resources consumed counters may have stopped still be incrementing, the</u>	
2460	<u>server/device moves the job from the <b>processing</b> state to -while the job is in the <b>canceled</b></u>	
2461	<u>or <b>aborted</b> job states.</u>	
2462		
2463	<b>serviceOffLine</b>	<b>0x40000</b>
2464	<u>The service or document transform is off-line and accepting no jobs. All <b>pending</b> jobs are</u>	
2465	<u>put into the <b>pendingHeld</b> state. This situation could be true if the service's or document</u>	
2466	<u>transform's input is impaired or broken.</u>	
2467		
2468	<b>jobCompletedSuccessfully</b>	<b>0x80000</b>
2469	The job completed successfully.	
2470		
2471	<b>jobCompletedWithWarnings</b>	<b>0x100000</b>
2472	The job completed with warnings.	
2473		
2474	<b>jobCompletedWithErrors</b>	<b>0x200000</b>
2475	The job completed with errors (and possibly warnings too).	
2476		
2477		
2478	The following additional job state reasons have been added to represent job states that are in	
2479	ISO DPA[iso-dpa] and other job submission protocols:	

2480  
 2481 | **jobPaused** **0x400000**  
 2482     The job has been indefinitely suspended by a client issuing an operation to suspend the job  
 2483     so that other jobs may proceed using the same devices. The client MAY issue an  
 2484     operation to resume the paused job at any time, in which case the agent SHALL remove  
 2485     the **jobPaused** values from the job's **jmJobStateReasons1** object and the job is eventually  
 2486     resumed at or near the point where the job was paused.  
 2487

2488 | **jobInterrupted** **0x800000**  
 2489     The job has been interrupted while processing by a client issuing an operation that  
 2490     specifies another job to be run instead of the current job. The server or device will  
 2491     automatically resume the interrupted job when the interrupting job completes.  
 2492

2493 | **jobRetained** **0x1000000**  
 2494     The job is being retained by the server or device with all of the job's document data (and  
 2495     submitted resources, such as fonts, logos, and forms, if any). Thus a client could issue an  
 2496     operation to the server or device to either (1) re-do the job (or a copy of the job) on the  
 2497     same server or device or (2) resubmit the job to another server or device. When a client  
 2498     could no longer re-do/resubmit the job, such as after the document data has been  
 2499     discarded, the agent SHALL remove the **jobRetained** value from the  
 2500     **jmJobStateReasons1** object."  
 2501 REFERENCE  
 2502     "These bit definitions are the equivalent of a type 2 enum except that combinations of bits may  
 2503     be used together. See section 3.6.1.2. The remaining bits are reserved for future  
 2504     standardization and/or registration."  
 2505

2506 SYNTAX   **INTEGER(0..2147483647)** -- 31 bits, all but sign bit  
 2507  
 2508  
 2509  
 2510  
 2511

2512 **JmJobStateReasons2TC ::= TEXTUAL-CONVENTION**  
 2513   STATUS   current  
 2514   DESCRIPTION  
 2515     "This textual-convention is used with the **jobStateReasons2** attribute to provides additional  
 2516     information regarding the **jmJobState** object. See the description under  
 2517     **JmJobStateReasons1TC** for additional information that applies to all reasons.  
 2518  
 2519     The following standard values are defined (in hexadecimal) as *powers of two*, since multiple  
 2520     values may be used at the same time:  
 2521

2522   **cascaded** **0x1**  
 2523     An outbound gateway has transmitted all of the job's job and document attributes and data  
 2524     to another spooling system.  
 2525

2526   **deletedByAdministrator** **0x2**  
 2527     The administrator has deleted the job.  
 2528

2529	<b>discardTimeArrived</b>	<b>0x4</b>
2530	The job has been deleted due to the fact that the time specified by the job's job-discard-time attribute has arrived.	
2531		
2532		
2533	<b>postProcessingFailed</b>	<b>0x8</b>
2534	The post-processing agent failed while trying to log accounting attributes for the job; therefore the job has been placed into the completed state with the <b>jobRetained</b>	
2535	<b>jmJobStateReasons1</b> object value for a system-defined period of time, so the	
2536	administrator can examine it, resubmit it, etc.	
2537		
2538		
2539	<b><u>jobTransforming</u></b>	<b><u>0x10</u></b>
2540	<u>The server/device is interpreting document data and producing another electronic</u>	
2541	<u>representation.</u>	
2542		
2543	<b><del>submissionInterrupted</del></b>	<b><del>0x10</del></b>
2544	<del>Indicates that the job was not completely submitted for some unforeseen reason, such as:</del>	
2545	<del>(1) the server has crashed before the job was closed by the client, (2) the server or the</del>	
2546	<del>document transfer method has crashed in some non-recoverable way before the document</del>	
2547	<del>data was entirely transferred to the server, (3) the client crashed or failed to close the job</del>	
2548	<del>before the time-out period.</del>	
2549		
2550	<b>maxJobFaultCountExceeded</b>	<b>0x20</b>
2551	The job has faulted several times and has exceeded the administratively defined fault count	
2552	limit.	
2553		
2554	<b>devicesNeedAttentionTimeOut</b>	<b>0x40</b>
2555	One or more document transforms that the job is using needs human intervention in order	
2556	for the job to make progress, but the human intervention did not occur within the site-	
2557	settable time-out value.	
2558		
2559	<b>needsKeyOperatorTimeOut</b>	<b>0x80</b>
2560	One or more devices or document transforms that the job is using need a specially trained	
2561	operator (who may need a key to unlock the device and gain access) in order for the job to	
2562	make progress, but the key operator intervention did not occur within the site-settable	
2563	time-out value.	
2564		
2565	<b>jobStartWaitTimeOut</b>	<b>0x100</b>
2566	The server/device has stopped the job at the beginning of processing to await human	
2567	action, such as installing a special cartridge or special non-standard media, but the job was	
2568	not resumed within the site-settable time-out value and the server/device has transitioned	
2569	the job to the <b>pendingHeld</b> state.	
2570		
2571	<b>jobEndWaitTimeOut</b>	<b>0x200</b>
2572	The server/device has stopped the job at the end of processing to await human action,	
2573	such as removing a special cartridge or restoring standard media, but the job was not	
2574	resumed within the site-settable time-out value and the server/device has transitioned the	
2575	job to the completed state.	
2576		



2577	<b>jobPasswordWaitTimeOut</b>	<b>0x400</b>
2578	The server/device has stopped the job at the beginning of processing to await input of the	
2579	job's password, but the password was not received within the site-settable time-out value.	
2580		
2581	<b>deviceTimedOut</b>	<b>0x800</b>
2582	A device that the job was using has not responded in a period specified by the device's	
2583	site-settable attribute.	
2584		
2585	<b>connectingToDeviceTimeOut</b>	<b>0x1000</b>
2586	The server is attempting to connect to one or more devices which may be dial-up, polled,	
2587	or queued, and so may be busy with traffic from other systems, but server was unable to	
2588	connect to the device within the site-settable time-out value.	
2589		
2590	<b>transferring</b>	<b>0x2000</b>
2591	The job is being transferred to a down stream server or <u>downstream</u> device.	
2592		
2593	<b>queuedInDevice</b>	<b>0x4000</b>
2594	The <u>server/device has</u> <del>job has been</del> <u>queued the job</u> in a down stream server or <u>downstream</u>	
2595	device.	
2596		
2597	<b><u>jobQueued</u></b>	<b><u>0x8000</u></b>
2598	<u>The server/device has queued the document data.</u>	
2599		
2600	<b>jobCleanup</b>	<b>0x108000</b>
2601	The server/device is performing cleanup activity as part of ending normal processing.	
2602		
2603	<b>jobPasswordWait</b>	<b>0x20000</b>
2604	The server/device has selected the job to be next to process, but instead of assigning	
2605	resources and starting the job processing, the server/device has transitioned the job to the	
2606	<b>pendingHeld</b> state to await entry of a password (and dispatched another job, if there is	
2607	one).	
2608		
2609	<b>validating</b>	<b>0x40000</b>
2610	The server/device is validating the job <i>after</i> accepting the job.	
2611		
2612	<b>queueHeld</b>	<b>0x80000</b>
2613	The operator has held the entire job set or queue.	
2614		
2615	<b>jobProofWait</b>	<b>0x100000</b>
2616	The job has produced a single proof copy and is in the <b>pendingHeld</b> state waiting for the	
2617	requester to issue an operation to release the job to print normally, obeying any job and	
2618	document copy attributes that were originally submitted.	
2619		
2620	<b>heldForDiagnostics</b>	<b>0x200000</b>
2621	The system is running intrusive diagnostics, so that all jobs are being held.	
2622		
2623	<b><del>serviceOffLine</del></b>	<b><del>0x400000</del></b>
2624	<del>The service/document transform is off-line and accepting no jobs. All pending jobs are put</del>	
2625	<del>into the pendingHeld state. This could be true if its input is impaired or broken.</del>	

2626  
 2627 **noSpaceOnServer** **0x800000**  
 2628 There is no room on the server to store all of the job.  
 2629  
 2630 **pinRequired** **0x1000000**  
 2631 The System Administrator settable device policy is (1) to require PINs, and (2) to hold  
 2632 jobs that do not have a pin supplied as an input parameter when the job was created.  
 2633  
 2634 **exceededAccountLimit** **0x2000000**  
 2635 The account for which this job is drawn has exceeded its limit. This condition SHOULD  
 2636 be detected before the job is scheduled so that the user does not wait until his/her job is  
 2637 scheduled only to find that the account is overdrawn. This condition MAY also occur  
 2638 while the job is processing either as processing begins or part way through processing.  
 2639  
 2640 **heldForRetry** **0x4000000**  
 2641 The job encountered some errors that the server/device could not recover from with its  
 2642 normal retry procedures, but the error might not be encountered if the job is processed  
 2643 again in the future. Example cases are phone number busy or remote file system in-  
 2644 accessible. For such a situation, the server/device SHALL transition the job from the  
 2645 **processing** to the **pendingHeld**, rather than to the **aborted** state.  
 2646  
 2647 The following values are from the X/Open PSIS draft standard:  
 2648  
 2649 **canceledByShutdown** **0x8000000**  
 2650 The job was canceled because the server or device was shutdown before completing the  
 2651 job.  
 2652  
 2653 **deviceUnavailable** **0x10000000**  
 2654 This job was aborted by the system because the device is currently unable to accept jobs.  
 2655  
 2656 **wrongDevice** **0x20000000**  
 2657 This job was aborted by the system because the device is unable to handle this particular  
 2658 job; the spooler SHOULD try another device or the user should submit the job to another  
 2659 device.  
 2660  
 2661 **badJob** **0x40000000**  
 2662 This job was aborted by the system because this job has a major problem, such as an ill-  
 2663 formed PDL; the spooler SHOULD not even try another device. "  
 2664 REFERENCE  
 2665 "These bit definitions are the equivalent of a type 2 enum except that combinations of them may  
 2666 be used together. See section 3.6.1.2. See the description under **JmJobStateReasons1TC** and  
 2667 the **jobStateReasons2** attribute."  
 2668  
 2669 SYNTAX **INTEGER(0..2147483647)** -- 31 bits, all but sign bit  
 2670  
 2671  
 2672  
 2673  
 2674

2675  
 2676 **JmJobStateReasons3TC** ::= TEXTUAL-CONVENTION  
 2677     STATUS     current  
 2678     DESCRIPTION  
 2679         "This textual-convention is used with the **jobStateReasons3** attribute to provides additional  
 2680         information regarding the **jmJobState** object. See the description under  
 2681         **JmJobStateReasons1TC** for additional information that applies to all reasons.  
 2682  
 2683         The following standard values are defined (in hexadecimal) as *powers of two*, since multiple  
 2684         values may be used at the same time:  
 2685  
 2686         **jobInterruptedByDeviceFailure**                     **0x1**  
 2687             A device or the print system software that the job was using has failed while the job was  
 2688             processing. The server or device is keeping the job in the **pendingHeld** state until an  
 2689             operator can determine what to do with the job."  
 2690     REFERENCE  
 2691         "These bit definitions are the equivalent of a type 2 enum except that combinations of them may  
 2692         be used together. See section 3.6.1.2. The remaining bits are reserved for future  
 2693         standardization and/or registration. See the description under **JmJobStateReasons1TC** and the  
 2694         **jobStateReasons3** attribute."  
 2695     SYNTAX     **INTEGER(0..2147483647)** -- 31 bits, all but sign bit  
 2696  
 2697  
 2698  
 2699  
 2700  
 2701 **JmJobStateReasons4TC** ::= TEXTUAL-CONVENTION  
 2702     STATUS     current  
 2703     DESCRIPTION  
 2704         "This textual-convention is used in the **jobStateReasons4** attribute to provides additional  
 2705         information regarding the **jmJobState** object. See the description under  
 2706         **JmJobStateReasons1TC** for additional information that applies to all reasons.  
 2707  
 2708         The following standard values are defined (in hexadecimal) as *powers of two*, since multiple  
 2709         values may be used at the same time:  
 2710  
 2711         none yet defined. These bits are reserved for future standardization and/or registration."  
 2712     REFERENCE  
 2713         "These bit definitions are the equivalent of a type 2 enum except that combinations of them may  
 2714         be used together. See section 3.6.1.2. See the description under **JmJobStateReasons1TC** and  
 2715         the **jobStateReasons4** attribute."  
 2716  
 2717     SYNTAX     **INTEGER(0..2147483647)** -- 31 bits, all but sign bit

```

2718
2719 jobmonMIBObjects OBJECT IDENTIFIER ::= { jobmonMIB 1 }
2720
2721 -- The General Group (MANDATORY)
2722
2723 -- The jmGeneralGroup consists entirely of the jmGeneralTable.
2724
2725 jmGeneral OBJECT IDENTIFIER ::= { jobmonMIBObjects 1 }
2726
2727 jmGeneralTable OBJECT-TYPE
2728     SYNTAX      SEQUENCE OF JmGeneralEntry
2729     MAX-ACCESS  not-accessible
2730     STATUS      current
2731     DESCRIPTION
2732         "The jmGeneralTable consists of information of a general nature that are per-job-set, but are
2733         not per-job. See Section 2 entitled 'Terminology and Job Model' for the definition of a job set."
2734     REFERENCE
2735         "The MANDATORY-GROUP macro specifies that this group is MANDATORY."
2736     ::= { jmGeneral 1 }
2737
2738 jmGeneralEntry OBJECT-TYPE
2739     SYNTAX      JmGeneralEntry
2740     MAX-ACCESS  not-accessible
2741     STATUS      current
2742     DESCRIPTION
2743         "Information about a job set (queue).
2744
2745         An entry SHALL exist in this table for each job set."
2746     INDEX { jmGeneralJobSetIndex }
2747     ::= { jmGeneralTable 1 }
2748
2749 JmGeneralEntry ::= SEQUENCE {
2750     jmGeneralJobSetIndex          Integer32(1..32767),
2751     jmGeneralNumberOfActiveJobs  Integer32(0..2147483647),
2752     jmGeneralOldestActiveJobIndex Integer32(0..2147483647),
2753     jmGeneralNewestActiveJobIndex Integer32(0..2147483647),
2754     jmGeneralJobPersistence      Integer32(15..2147483647),
2755     jmGeneralAttributePersistence Integer32(15..2147483647),
2756     jmGeneralJobSetName          JmUTF8StringTC(SIZE(0..63))
2757 }
2758
2759 jmGeneralJobSetIndex OBJECT-TYPE
2760     SYNTAX      Integer32(1..32767)
2761     MAX-ACCESS  not-accessible
2762     STATUS      current
2763     DESCRIPTION
2764         "A unique value for each job set in this MIB. The jmJobTable and jmAttributeTable tables
2765         have this same index as their primary index.
2766

```

2767 The value(s) of the **jmGeneralJobSetIndex** SHALL be persistent across power cycles, so that  
 2768 clients that have retained **jmGeneralJobSetIndex** values will access the same job sets upon  
 2769 subsequent power-up.  
 2770

2771 An implementation that has only one job set, such as a printer with a single queue, SHALL hard  
 2772 code this object with the value **1**."

2773 REFERENCE  
 2774 "See Section 2 entitled 'Terminology and Job Model' for the definition of a job set.  
 2775 Corresponds to the first index in **jmJobTable** and **jmAttributeTable**."  
 2776 ::= { jmGeneralEntry 1 }  
 2777

2778 **jmGeneralNumberOfActiveJobs** OBJECT-TYPE  
 2779 SYNTAX Integer32(0..2147483647)  
 2780 MAX-ACCESS read-only  
 2781 STATUS current  
 2782 DESCRIPTION  
 2783 "The current number of 'active' jobs in the **jmJobIDTable**, **jmJobTable**, and  
 2784 **jmAttributeTable**, i.e., the total number of jobs that are in the **pending**, **processing**, or  
 2785 **processingStopped** states. See the **JmJobStateTC** textual-convention for the exact  
 2786 specification of the semantics of the job states."  
 2787 ::= { jmGeneralEntry 2 }  
 2788

2789 **jmGeneralOldestActiveJobIndex** OBJECT-TYPE  
 2790 SYNTAX Integer32 (0..2147483647)  
 2791 MAX-ACCESS read-only  
 2792 STATUS current  
 2793 DESCRIPTION  
 2794 "The **jmJobIndex** of the oldest job that is still in one of the 'active' states (**pending**, **processing**,  
 2795 or **processingStopped**). In other words, the index of the 'active' job that has been in the job  
 2796 tables the longest.  
 2797  
 2798 If there are no active jobs, the agent SHALL set the value of this object to **0**."  
 2799 REFERENCE  
 2800 "See Section 3.2 entitled 'The Job Tables and the Oldest Active and Newest Active Indexes' for  
 2801 a description of the usage of this object."  
 2802 ::= { jmGeneralEntry 3 }  
 2803

2804 **jmGeneralNewestActiveJobIndex** OBJECT-TYPE  
 2805 SYNTAX Integer32 (0..2147483647)  
 2806 MAX-ACCESS read-only  
 2807 STATUS current  
 2808 DESCRIPTION  
 2809 "The **jmJobIndex** of the newest job that is in one of the 'active' states (**pending**, **processing**, or  
 2810 **processingStopped**). In other words, the index of the 'active' job that has been most recently  
 2811 added to the **job tables**.  
 2812  
 2813 When all jobs become 'inactive', i.e., enter the **pendingHeld**, **completed**, **canceled**, or **aborted**  
 2814 states, the agent SHALL set the value of this object to **0**."  
 2815 REFERENCE

2816 "See Section 3.2 entitled 'The Job Tables and the Oldest Active and Newest Active Indexes' for  
 2817 a description of the usage of this object."  
 2818 ::= { jmGeneralEntry 4 }  
 2819

**jmGeneralJobPersistence** OBJECT-TYPE  
 2820 SYNTAX Integer32(15..2147483647)  
 2821 UNITS "seconds"  
 2822 MAX-ACCESS read-only  
 2823 STATUS current  
 2824 DESCRIPTION  
 2825 "The minimum time in seconds for this instance of the Job Set that an entry SHALL remain in  
 2826 the **jmJobIDTable** and **jmJobTable** after **processing** has *completed*, i.e., the minimum time in  
 2827 seconds starting when the job enters the **completed**, **canceled**, or **aborted** state.  
 2828  
 2829 ~~Configuring this object is implementation-dependent. Depending on implementation, the value of~~  
 2830 ~~this object MAY be either: (1) set by the system administrator by means outside this~~  
 2831 ~~specification or (2) fixed by the implementation.~~  
 2832  
 2833 This value SHALL be equal to or greater than the value of **jmGeneralAttributePersistence**.  
 2834 This value SHOULD be at least 60 which gives a monitoring application one minute in which to  
 2835 poll for job data."  
 2836 DEFVAL { 60 } -- one minute  
 2837 ::= { jmGeneralEntry 5 }  
 2838  
 2839

**jmGeneralAttributePersistence** OBJECT-TYPE  
 2840 SYNTAX Integer32(15..2147483647)  
 2841 UNITS "seconds"  
 2842 MAX-ACCESS read-only  
 2843 STATUS current  
 2844 DESCRIPTION  
 2845 "The minimum time in seconds for this instance of the Job Set that an entry SHALL remain in  
 2846 the **jmAttributeTable** after **processing** has *completed* , i.e., the time in seconds starting when  
 2847 the job enters the **completed**, **canceled**, or **aborted** state.  
 2848  
 2849 ~~Configuring this object is implementation-dependent. Depending on implementation, the value of~~  
 2850 ~~this object MAY be either (1) set by the system administrator by means outside this specification~~  
 2851 ~~or MAY be (2) fixed by the implementation.~~  
 2852  
 2853 This value SHOULD be at least 60 which gives a monitoring application one minute in which to  
 2854 poll for job data."  
 2855 DEFVAL { 60 } -- one minute  
 2856 ::= { jmGeneralEntry 6 }  
 2857  
 2858

**jmGeneralJobSetName** OBJECT-TYPE  
 2859 SYNTAX JmUTF8StringTC(SIZE(0..63))  
 2860 MAX-ACCESS read-only  
 2861 STATUS current  
 2862 DESCRIPTION  
 2863

2864 "The human readable name of this job set assigned by the system administrator (by means  
 2865 outside of this MIB). Typically, this name SHOULD be the name of the job queue. If a server  
 2866 or device has only a single job set, this object can be the administratively assigned name of the  
 2867 server or device itself. This name does not need to be unique, though each job set in a single  
 2868 Job Monitoring MIB SHOULD have distinct names.  
 2869

2870 NOTE - The purpose of this object is to help the user of the job monitoring application  
 2871 distinguish between several job sets in implementations that support more than one job set."  
 2872 REFERENCE  
 2873 "See the OBJECT compliance macro for the minimum maximum length required for  
 2874 conformance."  
 2875 ::= { jmGeneralEntry 7 }  
 2876  
 2877  
 2878  
 2879  
 2880  
 2881 -- The Job ID Group (MANDATORY)  
 2882  
 2883 -- The **jmJobIDGroup** consists entirely of the **jmJobIDTable**.  
 2884  
 2885 jmJobID OBJECT IDENTIFIER ::= { jobmonMIBObjects 2 }  
 2886  
 2887 jmJobIDTable OBJECT-TYPE  
 2888 SYNTAX SEQUENCE OF JmJobIDEntry  
 2889 MAX-ACCESS not-accessible  
 2890 STATUS current  
 2891 DESCRIPTION  
 2892 "The **jmJobIDTable** provides a correspondence map (1) between the job submission ID that a  
 2893 client uses to refer to a job and (2) the **jmGeneralJobSetIndex** and **jmJobIndex** that the Job  
 2894 Monitoring MIB agent assigned to the job and that are used to access the job in all of the other  
 2895 tables in the MIB. If a monitoring application already knows the **jmGeneralJobSetIndex** and  
 2896 the **jmJobIndex** of the job it is querying, that application NEED NOT use the **jmJobIDTable**."  
 2897 REFERENCE  
 2898 "The MANDATORY-GROUP macro specifies that this group is MANDATORY."  
 2899 ::= { jmJobID 1 }  
 2900  
 2901 jmJobIDEntry OBJECT-TYPE  
 2902 SYNTAX JmJobIDEntry  
 2903 MAX-ACCESS not-accessible  
 2904 STATUS current  
 2905 DESCRIPTION  
 2906 "The map from (1) the **jmJobSubmissionID** to (2) the **jmGeneralJobSetIndex** and  
 2907 **jmJobIndex**.  
 2908  
 2909 An entry SHALL exist in this table for each job currently known to the agent for all job sets and  
 2910 job states. Each job SHALL appear in one and only one job set."  
 2911 INDEX { **jmJobSubmissionID** }  
 2912 ::= { jmJobIDTable 1 }

2913  
 2914 JmJobIDEntry ::= SEQUENCE {  
 2915     **jmJobSubmissionID**                     **OCTET STRING(SIZE(48)),**  
 2916     **jmJobIDJobSetIndex**                 **Integer32(1..32767),**  
 2917     **jmJobIDJobIndex**                     **Integer32(1..2147483647)**  
 2918 }  
 2919  
 2920 **jmJobSubmissionID** OBJECT-TYPE  
 2921     SYNTAX     **OCTET STRING(SIZE(48))**  
 2922     MAX-ACCESS not-accessible  
 2923     STATUS     current  
 2924     DESCRIPTION  
 2925         "A quasi-unique 48-octet fixed-length string ID which identifies the job within a particular  
 2926         client-server environment. There are multiple formats for the **jmJobSubmissionID**. Each  
 2927         format SHALL be uniquely identified. See the JmJobSubmissionIDTypeTC textual convention.  
 2928         Each format SHALL be registered using the procedures of a type 2 enum. See section 3.6.3  
 2929         entitled: 'IANA Registration of Job Submission Id Formats'.  
 2930  
 2931         If the requester (client or server) does not supply a job submission ID in the job submission  
 2932         protocol, then the recipient (server or device) SHALL assign a job submission ID using any of  
 2933         the standard formats that have been reserved ~~for~~ agents and adding the final 8 octets to  
 2934         distinguish the ID from others submitted from the same requester.  
 2935  
 2936         The monitoring application, whether in the client or running separately, MAY use the job  
 2937         submission ID to help identify which **jmJobIndex** was assigned by the agent, i.e., in which row  
 2938         the job information is in the other tables.  
 2939  
 2940         NOTE - fixed-length is used so that a management application can use a shortened GetNext  
 2941         varbind (in SNMPv1 and SNMPv2) in order to get the next submission ID, disregarding the  
 2942         remainder of the ID in order to access jobs independent of the trailing identifier part, e.g., to get  
 2943         all jobs submitted by a particular **jmJobOwner** or submitted from a particular MAC address."  
 2944     REFERENCE  
 2945         "See the **JmJobSubmissionIDTypeTC** textual convention.  
 2946         See APPENDIX B - Support of the Job Submission ID in Job Submission Protocols."  
 2947     ::= { jmJobIDEntry 1 }  
 2948  
 2949 **jmJobIDJobSetIndex** OBJECT-TYPE  
 2950     SYNTAX     **Integer32(1..32767)**  
 2951     MAX-ACCESS read-only  
 2952     STATUS     current  
 2953     DESCRIPTION  
 2954         "This object contains the value of the **jmGeneralJobSetIndex** for the job with the  
 2955         **jmJobSubmissionID** value, i.e., the job set index of the job set in which the job was placed  
 2956         when that server or device accepted the job. This 16-bit value in combination with the  
 2957         **jmJobIDJobIndex** value permits the management application to access the other tables to  
 2958         obtain the job-specific objects for this job."  
 2959     REFERENCE  
 2960         "See **jmGeneralJobSetIndex** in the **jmGeneralTable**."  
 2961     ::= { jmJobIDEntry 2 }



```

2962
2963 jmJobIDJobIndex OBJECT-TYPE
2964     SYNTAX      Integer32(1..2147483647)
2965     MAX-ACCESS  read-only
2966     STATUS      current
2967     DESCRIPTION
2968         "This object contains the value of the jmJobIndex for the job with the jmJobSubmissionID
2969         value, i.e., the job index for the job when the server or device accepted the job. This value, in
2970         combination with the jmJobIDJobSetIndex value, permits the management application to
2971         access the other tables to obtain the job-specific objects for this job."
2972     REFERENCE
2973         "See jmJobIndex in the jmJobTable."
2974     ::= { jmJobIDEntry 3 }
2975
2976
2977
2978
2979 -- The Job Group (MANDATORY)
2980
2981 -- The jmJobGroup consists entirely of the jmJobTable.
2982
2983 jmJob OBJECT IDENTIFIER ::= { jobmonMIBObjects 3 }
2984
2985 jmJobTable OBJECT-TYPE
2986     SYNTAX      SEQUENCE OF JmJobEntry
2987     MAX-ACCESS  not-accessible
2988     STATUS      current
2989     DESCRIPTION
2990         "The jmJobTable consists of basic job state and status information for each job in a job set that
2991         (1) monitoring applications need to be able to access in a single SNMP Get operation, (2) that
2992         have a single value per job, and (3) that SHALL always be implemented."
2993     REFERENCE
2994         "The MANDATORY-GROUP macro specifies that this group is MANDATORY."
2995     ::= { jmJob 1 }
2996
2997 jmJobEntry OBJECT-TYPE
2998     SYNTAX      JmJobEntry
2999     MAX-ACCESS  not-accessible
3000     STATUS      current
3001     DESCRIPTION
3002         "Basic per-job state and status information.
3003
3004         An entry SHALL exist in this table for each job, no matter what the state of the job is. Each job
3005         SHALL appear in one and only one job set."
3006     REFERENCE
3007         "See Section 3.2 entitled 'The Job Tables'."
3008     INDEX { jmGeneralJobSetIndex, jmJobIndex }
3009     ::= { jmJobTable 1 }
3010

```

```

3011 JmJobEntry ::= SEQUENCE {
3012     jmJobIndex          Integer32(1..2147483647),
3013     jmJobState          JmJobStateTC,
3014     jmJobStateReasons1 JmJobStateReasons1TC,
3015     jmNumberOfInterveningJobs Integer32(-2..2147483647),
3016     jmJobKOctetsRequested Integer32(-2..2147483647),
3017     jmJobKOctetsProcessed Integer32(-2..2147483647),
3018     jmJobImpressionsRequested Integer32(-2..2147483647),
3019     jmJobImpressionsCompleted Integer32(-2..2147483647),
3020     jmJobOwner          JmJobStringTC(SIZE(0..63))
3021 }
3022
3023 jmJobIndex OBJECT-TYPE
3024     SYNTAX      Integer32(1..2147483647)
3025     MAX-ACCESS  not-accessible
3026     STATUS      current
3027     DESCRIPTION
3028         "The sequential, monotonically increasing identifier index for the job generated by the server or
3029         device when that server or device accepted the job. This index value permits the management
3030         application to access the other tables to obtain the job-specific row entries."
3031     REFERENCE
3032         "See Section 3.2 entitled 'The Job Tables and the Oldest Active and Newest Active Indexes'.
3033         See Section 3.4 entitled 'Job Identification'.
3034         See also jmGeneralNewestActiveJobIndex for the largest value of jmJobIndex.
3035         See JmJobSubmissionTypeTC for a limit on the size of this index if the agent represents it as
3036         an 8-digit decimal number."
3037     ::= { jmJobEntry 1 }
3038
3039 jmJobState OBJECT-TYPE
3040     SYNTAX      JmJobStateTC
3041     MAX-ACCESS  read-only
3042     STATUS      current
3043     DESCRIPTION
3044         "The current state of the job (pending, processing, completed, etc.). Agents SHALL
3045         implement only those states which are appropriate for the particular implementation. However,
3046         management applications SHALL be prepared to receive all the standard job states.
3047
3048         The final value for this object SHALL be one of: completed, canceled, or aborted. The
3049         minimum length of time that the agent SHALL maintain MIB data for a job in the completed,
3050         canceled, or aborted state before removing the job data from the jmJobIDTable and
3051         jmJobTable is specified by the value of the jmGeneralJobPersistence object."
3052     ::= { jmJobEntry 2 }
3053
3054 jmJobStateReasons1 OBJECT-TYPE
3055     SYNTAX      JmJobStateReasons1TC
3056     MAX-ACCESS  read-only
3057     STATUS      current
3058     DESCRIPTION

```

3059 "Additional information about the job's current state, i.e., information that augments the value of  
3060 the job's **jmJobState** object.  
3061  
3062 Implementation of any reason values is OPTIONAL, but an agent SHOULD return any reason  
3063 information available. These values MAY be used with any job state or states for which the  
3064 reason makes sense. Since the Job State Reasons will be more dynamic than the Job State, it is  
3065 recommended that a job monitoring application read this object every time **jmJobState** is  
3066 read. Furthermore, when implemented as with any MIB data, the agent SHALL return these  
3067 values when the reason applies and SHALL NOT return them when the reason no longer applies  
3068 whether the value of the job's **jmJobState** object changed or not. When the agent cannot  
3069 provide a reason for the current state of the job, the agent SHALL set the value of the  
3070 **jmJobStateReasons1** object and **jobStateReasonsN** attributes SHALL be 0."  
3071 REFERENCE  
3072 "The **jobStateReasonsN** ( $N=2..4$ ) attributes provide further additional information about the  
3073 job's current state."  
3074 ::= { jmJobEntry 3 }  
3075  
3076 **jmNumberOfInterveningJobs** OBJECT-TYPE  
3077 SYNTAX Integer32(-2..2147483647)  
3078 MAX-ACCESS read-only  
3079 STATUS current  
3080 DESCRIPTION  
3081 "The number of jobs that are expected to complete ~~being processed~~ *before* this job has  
3082 completed ~~being processed~~ according to the implementation's queuing algorithm, if no other  
3083 jobs were to be submitted. In other words, this value is the job's queue position. The agent  
3084 SHALL return a value of 0 for this attribute when the job is the next job to complete processing  
3085 (or has completed processing)."  
3086 ::= { jmJobEntry 4 }  
3087  
3088 **jmJobKOctetsRequested** OBJECT-TYPE  
3089 SYNTAX Integer32(-2..2147483647)  
3090 MAX-ACCESS read-only  
3091 STATUS current  
3092 DESCRIPTION  
3093 "The total size in K (1024) octets of the document(s) being requested to be processed in the job.  
3094 The agent SHALL round the actual number of octets up to the next highest K. Thus 0 octets  
3095 SHALL be represented as '0', 1-1024 octets SHALL be represented as '1', 1025-2048 SHALL  
3096 be represented as '2', etc.  
3097  
3098 In computing this value, the server/device SHALL *not* include the multiplicative factors  
3099 contributed by (1) the number of document copies, and (2) the number of job copies,  
3100 independent of whether the device can process multiple copies of the job or document without  
3101 making multiple passes over the job or document data and independent of whether the output is  
3102 collated or not. Thus the server/device computation is independent of the implementation."  
3103 ::= { jmJobEntry 5 }  
3104  
3105 **jmJobKOctetsProcessed** OBJECT-TYPE  
3106 SYNTAX Integer32(-2..2147483647)  
3107 MAX-ACCESS read-only

3108 STATUS current  
3109 DESCRIPTION  
3110 "The current number of octets processed by the server or device measured in units of K (1024)  
3111 octets. The agent SHALL round the actual number of octets processed up to the next higher K.  
3112 Thus 0 octets SHALL be represented as '0', 1-1024 octets SHALL be represented as '1', 1025-  
3113 2048 octets SHALL be '2', etc. For printing devices, this value is the number interpreted by the  
3114 page description language interpreter rather than what has been marked on media.  
3115  
3116 For implementations where multiple copies are produced by the interpreter with only a single  
3117 pass over the data, the final value SHALL be equal to the value of the  
3118 **jmJobKOctetsRequested** object. For implementations where multiple copies are produced by  
3119 the interpreter by processing the data for each copy, the final value SHALL be a multiple of the  
3120 value of the **jmJobKOctetsRequested** object.  
3121  
3122 NOTE - See the **impressionsCompletedCurrentCopy** and **pagesCompletedCurrentCopy**  
3123 attributes for attributes that are reset on each document copy.  
3124  
3125 NOTE - The **jmJobKOctetsProcessed** object can be used with the **jmJobKOctetsRequested**  
3126 object to provide an indication of the relative progress of the job, provided that the  
3127 multiplicative factor is taken into account for some implementations of multiple copies."  
3128 ::= { jmJobEntry 6 }  
3129  
3130 **jmJobImpressionsRequested** OBJECT-TYPE  
3131 SYNTAX Integer32(-2..2147483647)  
3132 MAX-ACCESS read-only  
3133 STATUS current  
3134 DESCRIPTION  
3135 "The total size in number of impressions of the document(s) being requested by this job to  
3136 produce.  
3137  
3138 In computing this value, the server/device SHALL *not* include the multiplicative factors  
3139 contributed by (1) the number of document copies, and (2) the number of job copies,  
3140 independent of whether the device can process multiple copies of the job or document without  
3141 making multiple passes over the job or document data and independent of whether the output is  
3142 collated or not. Thus the server/device computation is independent of the implementation."  
3143 ::= { jmJobEntry 7 }  
3144  
3145 **jmJobImpressionsCompleted** OBJECT-TYPE  
3146 SYNTAX Integer32(-2..2147483647)  
3147 MAX-ACCESS read-only  
3148 STATUS current  
3149 DESCRIPTION  
3150 "The current number of impressions completed for this job so far. For printing devices, the  
3151 impressions completed includes interpreting, marking, and stacking the output. For other types  
3152 of job services, the number of impressions completed includes the number of impressions  
3153 processed.  
3154  
3155 For implementations where multiple copies are produced by the interpreter with only a single  
3156 pass over the data, the final value SHALL be equal to the value of the

3157 **jmJobImpressionsRequested** object. For implementations where multiple copies are produced  
 3158 by the interpreter by processing the data for each copy, the final value SHALL be a multiple of  
 3159 the value of the **jmJobImpressionsRequested** object.  
 3160

3161 NOTE - See the **impressionsCompletedCurrentCopy** and **pagesCompletedCurrentCopy**  
 3162 attributes for attributes that are reset on each document copy.  
 3163

3164 NOTE - The **jmJobImpressionsCompleted** object can be used with the  
 3165 **jmJobImpressionsRequested** object to provide an indication of the relative progress of the job,  
 3166 provided that the multiplicative factor is taken into account for some implementations of  
 3167 multiple copies."  
 3168 ::= { jmJobEntry 8 }  
 3169

3170 **jmJobOwner** OBJECT-TYPE

3171 SYNTAX **JmJobStringTC(SIZE(0..63))**

3172 MAX-ACCESS read-only

3173 STATUS current

3174 DESCRIPTION

3175 "The coded character set name of the user that submitted the job. The method of assigning this  
 3176 user name will be system and/or site specific but the method MUST insure that the name is  
 3177 unique to the network that is visible to the client and target device.  
 3178

3179 This value SHOULD be the *authenticated* name of the user submitting the job."  
 3180 REFERENCE

3181 "See the OBJECT compliance macro for the minimum maximum length required for  
 3182 conformance."  
 3183 ::= { jmJobEntry 9 }  
 3184  
 3185  
 3186  
 3187

3188 -- The Attribute Group (MANDATORY)

3189 -- The **jmAttributeGroup** consists entirely of the **jmAttributeTable**.

3190 --

3191 -- Implementation of the two objects in this group is MANDATORY.

3192 -- See Section 3.1 entitled 'Conformance Considerations'.

3193 -- An agent SHALL implement any attribute if (1) the server or device

3194 -- supports the functionality represented by the attribute and (2) the

3195 -- information is available to the agent.  
 3196  
 3197

3198 jmAttribute OBJECT IDENTIFIER ::= { jobmonMIBObjects 4 }  
 3199

3200 **jmAttributeTable** OBJECT-TYPE

3201 SYNTAX SEQUENCE OF JmAttributeEntry

3202 MAX-ACCESS not-accessible

3203 STATUS current

3204 DESCRIPTION

3205 "The **jmAttributeTable** SHALL contain attributes of the job and document(s) for each job in a  
3206 job set. Instead of allocating distinct objects for each attribute, each attribute is represented as a  
3207 separate row in the **jmAttributeTable**."  
3208 REFERENCE  
3209 "The MANDATORY-GROUP macro specifies that this group is MANDATORY. An agent  
3210 SHALL implement any attribute if (1) the server or device supports the functionality represented  
3211 by the attribute and (2) the information is available to the agent. "  
3212 ::= { **jmAttribute** 1 }

3213  
3214 **jmAttributeEntry** OBJECT-TYPE  
3215 SYNTAX JmAttributeEntry  
3216 MAX-ACCESS not-accessible  
3217 STATUS current  
3218 DESCRIPTION  
3219 "Attributes representing information about the job and document(s) or resources required and/or  
3220 consumed.  
3221  
3222 Each entry in the **jmAttributeTable** is a per-job entry with an extra index for each type of  
3223 attribute (**jmAttributeTypeIndex**) that a job can have and an additional index  
3224 (**jmAttributeInstanceIndex**) for those attributes that can have multiple instances per job. The  
3225 **jmAttributeTypeIndex** object SHALL contain an enum type that indicates the type of attribute  
3226 (see the **JmAttributeTypeTC** textual-convention). The value of the attribute SHALL be  
3227 represented in either the **jmAttributeValueAsInteger** or **jmAttributeValueAsOctets** objects,  
3228 and/or both, as specified in the **JmAttributeTypeTC** textual-convention.  
3229  
3230 The agent SHALL create rows in the **jmAttributeTable** as the server or device is able to  
3231 discover the attributes either from the job submission protocol itself or from the document PDL.  
3232 As the documents are interpreted, the interpreter MAY discover additional attributes and so the  
3233 agent adds additional rows to this table. As the attributes that represent resources are actually  
3234 consumed, the usage counter contained in the **jmAttributeValueAsInteger** object is  
3235 incremented according to the units indicated in the description of the **JmAttributeTypeTC**  
3236 enum.  
3237  
3238 The agent SHALL maintain each row in the **jmJobTable** for at least the minimum time after a  
3239 job completes as specified by the **jmGeneralAttributePersistence** object.  
3240  
3241 Zero or more entries SHALL exist in this table for each job in a job set."  
3242 REFERENCE  
3243 "See Section 3.3 entitled 'The Attribute Mechanism' for a description of the **jmAttributeTable**."  
3244 INDEX { **jmGeneralJobSetIndex**, **jmJobIndex**, **jmAttributeTypeIndex**,  
3245 **jmAttributeInstanceIndex** }  
3246 ::= { jmAttributeTable 1 }

3247  
3248 **JmAttributeEntry** ::= SEQUENCE {  
3249 **jmAttributeTypeIndex** **JmAttributeTypeTC**,  
3250 **jmAttributeInstanceIndex** **Integer32(1..32767)**,  
3251 **jmAttributeValueAsInteger** **Integer32(-2..2147483647)**,  
3252 **jmAttributeValueAsOctets** **OCTET STRING(SIZE(0..63))**  
3253 }

3254  
3255 **jmAttributeTypeIndex** OBJECT-TYPE  
3256 SYNTAX **JmAttributeTypeTC**  
3257 MAX-ACCESS not-accessible  
3258 STATUS current  
3259 DESCRIPTION  
3260 "The type of attribute that this row entry represents.  
3261  
3262 The type MAY identify information about the job or document(s) or MAY identify a resource  
3263 required to process the job before the job start processing and/or consumed by the job as the job  
3264 is processed.  
3265  
3266 Examples of job attributes (i.e., apply to the job as a whole) that have only one instance per job  
3267 include: **jobCopiesRequested(90)**, **documentCopiesRequested(92)**,  
3268 **jobCopiesCompleted(91)**, **documentCopiesCompleted(93)**, while examples of job attributes  
3269 that may have more than one instance per job include: **documentFormatIndex(37)**, and  
3270 **documentFormat(38)**.  
3271  
3272 Examples of document attributes (one instance per document) include: **fileName(34)**, and  
3273 **documentName(35)**.  
3274  
3275 Examples of required and consumed resource attributes include: **pagesRequested(130)**,  
3276 **mediumRequested(170)**, **pagesCompleted(131)**, and **mediumConsumed(171)**, respectively."  
3277 ::= { jmAttributeEntry 1 }  
3278  
3279 **jmAttributeInstanceIndex** OBJECT-TYPE  
3280 SYNTAX **Integer32(1..32767)**  
3281 MAX-ACCESS not-accessible  
3282 STATUS current  
3283 DESCRIPTION  
3284 "A running 16-bit index of the attributes of the same type for each job. For those attributes with  
3285 only a single instance per job, this index value SHALL be **1**. For those attributes that are a  
3286 single value per document, the index value SHALL be the document number, starting with **1** for  
3287 the first document in the job. Jobs with only a single document SHALL use the index value of  
3288 **1**. For those attributes that can have multiple values per job or per document, such as  
3289 **documentFormatIndex(37)** or **documentFormat(38)**, the index SHALL be a running index  
3290 for the job as a whole, starting at **1**."  
3291 ::= { jmAttributeEntry 2 }  
3292  
3293 **jmAttributeValueAsInteger** OBJECT-TYPE  
3294 SYNTAX **Integer32(-2..2147483647)**  
3295 MAX-ACCESS read-only  
3296 STATUS current  
3297 DESCRIPTION  
3298 "The integer value of the attribute. The value of the attribute SHALL be represented as an  
3299 integer if the enum description in the **JmAttributeTypeTC** textual-convention definition has the  
3300 tag: 'INTEGER:'.  
3301

3302 Depending on the enum definition, this object value MAY be an integer, a counter, an index, or  
 3303 an enum, depending on the **jmAttributeTypeIndex** value. The units of this value are specified  
 3304 in the enum description.  
 3305

3306 For those attributes that are accumulating job consumption as the job is processed as specified in  
 3307 the **JmAttributeTypeTC** textual-convention, SHALL contain the final value after the job  
 3308 completes processing, i.e., this value SHALL indicate the total usage of this resource made by  
 3309 the job.  
 3310

3311 A monitoring application is able to copy this value to a suitable longer term storage for later  
 3312 processing as part of an accounting system.  
 3313

3314 Since the agent MAY add attributes representing resources to this table while the job is waiting  
 3315 to be processed or being processed, which can be a long time before any of the resources are  
 3316 actually used, the agent SHALL set the value of the **jmAttributeValueAsInteger** object to **0**  
 3317 for resources that the job has not yet consumed.  
 3318

3319 Attributes for which the concept of an integer value is meaningless, such as **fileName(34)**,  
 3320 **jobName**, and **processingMessage**, do *not* have the 'INTEGER:' tag in the  
 3321 **JmAttributeTypeTC** definition and so an agent SHALL always return a value of '-1' to indicate  
 3322 'other' for the value of the **jmAttributeValueAsInteger** object for these attributes.  
 3323

3324 For attributes which do have the 'INTEGER:' tag in the **JmAttributeTypeTC** definition, if the  
 3325 integer value is not (yet) known, the agent either (1) SHALL not materialize the row in the  
 3326 **jmAttributeTable** until the value is known or (2) SHALL return a '-2' to represent an  
 3327 'unknown' counting integer value, a '0' to represent an 'unknown' index value, and a '2' to  
 3328 represent an 'unknown(2)' enum value."  
 3329 ::= { jmAttributeEntry 3 }  
 3330

3331 **jmAttributeValueAsOctets** OBJECT-TYPE  
 3332 SYNTAX OCTET STRING(SIZE(0..63))  
 3333 MAX-ACCESS read-only  
 3334 STATUS current  
 3335 DESCRIPTION  
 3336 "The octet string value of the attribute. The value of the attribute SHALL be represented as an  
 3337 OCTET STRING if the enum description in the **JmAttributeTypeTC** textual-convention  
 3338 definition has the tag: 'OCTETS:'.  
 3339

3340 Depending on the enum definition, this object value MAY be a coded character set string (text),  
 3341 such as '**JmUTF8StringTC**', or a binary octet string, such as '**DateAndTime**'.  
 3342

3343 Attributes for which the concept of an octet string value is meaningless, such as  
 3344 **pagesCompleted**, do *not* have the tag 'OCTETS:' in the **JmAttributeTypeTC** definition and so  
 3345 the agent SHALL always return a zero length string for the value of the  
 3346 **jmAttributeValueAsOctets** object.  
 3347

3348 For attributes which do have the 'OCTETS:' tag in the **JmAttributeTypeTC** definition, if the  
 3349 OCTET STRING value is not (yet) known, the agent either SHALL not materialize the row in  
 3350 the **jmAttributeTable** until the value is known or SHALL return a zero-length string."



3351 ::= { jmAttributeEntry 4 }  
3352

```

3353 -- Notifications and Trapping
3354 -- Reserved for the future
3355
3356 jobmonMIBNotifications OBJECT IDENTIFIER ::= { jobmonMIB 2 }
3357
3358
3359
3360 -- Conformance Information
3361
3362 jmMIBConformance OBJECT IDENTIFIER ::= { jobmonMIB 3 }
3363
3364 -- compliance statements
3365 jmMIBCompliance MODULE-COMPLIANCE
3366     STATUS current
3367     DESCRIPTION
3368         "The compliance statement for agents that implement the
3369         job monitoring MIB."
3370     MODULE -- this module
3371     MANDATORY-GROUPS {
3372         jmGeneralGroup, jmJobIDGroup, jmJobGroup, jmAttributeGroup }
3373
3374     OBJECT jmGeneralJobSetName
3375     SYNTAX JmUTF8StringTC (SIZE(0..8))
3376     DESCRIPTION
3377         "Only 8 octets maximum string length NEED be supported by the agent."
3378
3379     OBJECT jmJobOwner
3380     SYNTAX JmJobStringTC (SIZE(0..16))
3381     DESCRIPTION
3382         "Only 16 octets maximum string length NEED be supported by the agent."
3383
3384 -- There are no CONDITIONALLY MANDATORY or OPTIONAL groups.
3385
3386     ::= { jmMIBConformance 1 }
3387
3388 jmMIBGroups OBJECT IDENTIFIER ::= { jmMIBConformance 2 }
3389
3390 jmGeneralGroup OBJECT-GROUP
3391     OBJECTS {
3392         jmGeneralNumberOfActiveJobs, jmGeneralOldestActiveJobIndex,
3393         jmGeneralNewestActiveJobIndex, jmGeneralJobPersistence,
3394         jmGeneralAttributePersistence, jmGeneralJobSetName }
3395     STATUS current
3396     DESCRIPTION
3397         "The general group."
3398     ::= { jmMIBGroups 1 }
3399
3400 jmJobIDGroup OBJECT-GROUP
3401     OBJECTS {

```

```
3402         jmJobIDJobSetIndex, jmJobIDJobIndex }
3403     STATUS current
3404     DESCRIPTION
3405         "The job ID group."
3406     ::= { jmMIBGroups 2 }
3407
3408     jmJobGroup OBJECT-GROUP
3409     OBJECTS {
3410         jmJobState, jmJobStateReasons1, jmNumberOfInterveningJobs,
3411         jmJobKOctetsRequested, jmJobKOctetsProcessed, jmJobImpressionsRequested,
3412         jmJobImpressionsCompleted, jmJobOwner }
3413     STATUS current
3414     DESCRIPTION
3415         "The job group."
3416     ::= { jmMIBGroups 3 }
3417
3418     jmAttributeGroup OBJECT-GROUP
3419     OBJECTS {
3420         jmAttributeValueAsInteger, jmAttributeValueAsOctets }
3421     STATUS current
3422     DESCRIPTION
3423         "The attribute group."
3424     ::= { jmMIBGroups 4 }
3425
3426
3427     END
```

## 3428 **5. Appendix A - Implementing the Job Life Cycle**

3429 The job object has well-defined states and client operations that affect the transition between the  
3430 job states. Internal server and device actions also affect the transitions of the job between the job  
3431 states. These states and transitions are referred to as the job's *life cycle*.

3432 Not all implementations of job submission protocols have all of the states of the job model  
3433 specified here. The job model specified here is intended to be a superset of most implementations.  
3434 It is the purpose of the agent to map the particular implementation's job life cycle onto the one  
3435 specified here. The agent MAY omit any states not implemented. Only the **processing** and  
3436 **completed** states are required to be implemented by an agent. However, a conforming  
3437 management application SHALL be prepared to accept any of the states in the job life cycle  
3438 specified here, so that the management application can interoperate with any conforming agent.

3439 The job states are intended to be user visible. The agent SHALL make these states visible in the  
3440 MIB, but only for the subset of job states that the implementation has. Some implementations  
3441 MAY need to have sub-states of these user-visible states. The **jmJobStateReasons1** object and  
3442 the **jobStateReasonsN** ( $N=2..4$ ) attributes can be used to represent the sub-states of the jobs.

3443 Job states are intended to last a user-visible length of time in most implementations. However,  
3444 some jobs may pass through some states in zero time in some situations and/or in some  
3445 implementations.

3446 The job model does not specify how accounting and auditing is implemented, except to assume  
3447 that accounting and auditing logs are separate from the job life cycle and last longer than job  
3448 entries in the MIB. Jobs in the **completed**, **aborted**, or **canceled** states are not logs, since jobs in  
3449 these states are accessible via SNMP protocol operations and SHALL be removed from the Job  
3450 Monitoring MIB tables after a site-settable or implementation-defined period of time. An  
3451 accounting application MAY copy accounting information incrementally to an accounting log as a  
3452 job processes, or MAY be copied while the job is in the **canceled**, **aborted**, or **completed** states,  
3453 depending on implementation. The same is true for auditing logs.

3454 **The jmJobState object specifies the standard job states. The normal job state transitions**  
3455 **are shown in the state transition diagram presented in Table 1.**

## 3456 **6. APPENDIX B - Support of the Job Submission ID in Job Submission** 3457 **Protocols**

3458 This appendix lists the job submission protocols that support the concept of a job  
3459 submission ID and indicates the attribute used in that job submission protocol.

## 3460 6.1 Hewlett-Packard's Printer Job Language (PJL)

3461 Hewlett-Packard's Printer Job Language provides job-level printer control and printer  
3462 status information to applications. The PJL JOB command is used at the beginning of a  
3463 print job and can include options applying only to that job. A PJL JOB command option  
3464 has been defined to facilitate passing the **JobSubmissionID** with the print job, as required  
3465 by the Job Monitoring MIB. The option is of the form:

```
3466         SUBMISSIONID = "id string"  
3467  
3468
```

3469 Where the "id string" is a string and SHALL be enclosed in double quotes. The format is  
3470 as described for the **jmJobSubmissionID** object.

3471 The entire PJL JOB command with the optional parameter would be of the form:

```
3472         @PJL JOB SUBMISSIONID = "id string"  
3473  
3474
```

3475 See "Printer Job Language Technical Reference Manual", part number 5021-0328, from  
3476 Hewlett-Packard for complete information on the PJL JOB command and the Printer Job  
3477 Language.

3478 NOTE - Some PJL implementations wrap a banner page as a PJL job around a job  
3479 submitted by a client. In this case, there will be two job submission ids. The outer one  
3480 being the one with the banner page and the inner one being the original user's job. The  
3481 agent SHALL use the last received job submission ID for the **jmJobSubmissionID** index,  
3482 so that the original user's job submission ID will be used, not the banner page job ID.

## 3483 6.2 ISO DPA

3484 The ISO 10175 Document Printing Application (DPA) protocol specifies the "**job-client-  
3485 id**" attribute that allows the client to supply a text string ID for each job.

## 3486 7. References

3487 [char-set policy] Harald Avelstrand, "IETF Policy on Character Sets and Language", June  
3488 1997. Latest draft: <draft-avelstrand-charset-policy-00.txt>

3489 [GB2312] GB 2312-1980, "Chinese People's Republic of China (PRC) mixed one byte  
3490 and two byte coded character set"

3491 [hr-mib] P. Grillo, S. Waldbusser, "Host Resources MIB", RFC 1514, September 1993

3492 [iana] J. Reynolds, and J. Postel, "Assigned Numbers", STD 2, RFC 1700, ISI, October  
3493 1994.

- 3494 [IANA-charsets] Coded Character Sets registered by IANA and assigned an enum value  
3495 for use in the **CodedCharSet** textual convention imported from the Printer MIB. See  
3496 <ftp://ftp.isi.edu/in-notes/iana/assignments/character-sets>
- 3497 [iana-media-types] IANA Registration of MIME media types (MIME content  
3498 types/subtypes). See <ftp://ftp.isi.edu/in-notes/iana/assignments/>
- 3499 [ISO 646] ISO/IEC 646:1991, "Information technology -- ISO 7-bit coded character set  
3500 for information interchange", JTC1/SC2.
- 3501 [ISO 8859] ISO/IEC 8859-1:1987, "Information technology -- 8-bit single byte coded  
3502 graphic character sets - Part 1: Latin alphabet No. 1, JTC1/SC2."
- 3503 [ISO 2022] ISO/IEC 2022:1994 - "Information technology -- Character code structure  
3504 and extension techniques", JTC1/SC2.
- 3505 [ISO-10646] ISO/IEC 10646-1:1993, "Information technology -- Universal Multiple-  
3506 Octet Coded Character Set (UCS) - Part 1: Architecture and Basic Multilingual Plane,  
3507 JTC1/SC2.
- 3508 [iso-dpa] ISO/IEC 10175 Document Printing Application (DPA). See  
3509 <ftp://ftp.pwg.org/pub/pwg/dpa/>
- 3510 [ipp-model] Internet Printing Protocol (IPP), work in progress on the IETF standards  
3511 track. See **draft-ietf-ipp-model-01.txt**. See also <http://www.pwg.org/ipp/index.html>
- 3512 [JIS X0208] JIS X0208-1990, "Japanese two byte coded character set."
- 3513 [mib-II] MIB-II, RFC 1213.
- 3514 [print-mib] The Printer MIB - RFC 1759, proposed IETF standard. Also an Internet-  
3515 Draft on the standards track as a draft standard: **draft-ietf-printmib-mib-info-02.txt**
- 3516 [req-words] S. Bradner, "Keywords for use in RFCs to Indicate Requirement Levels",  
3517 RFC 2119, March 1997.
- 3518 [rfc 1738] Berners-Lee, T., Masinter, L., McCahill, M., "Uniform Resource Locators  
3519 (URL)", RFC 1738, December 1994.
- 3520 [rfc 2130] C. Weider, C. Preston, K. Simonsen, H. Alvestrand, R. Atkinson, M. Crispin,  
3521 and P. Svanberg, "The Report of the IAB Character Set Workshop held 29 Feb-1 March,  
3522 1997", April 1997, RFC 2130.
- 3523 [SMIv2-TC] J. Case, et al. "Textual Conventions for Version 2 of the Simple Network  
3524 Management Protocol (SNMPv2)", RFC 1903, January 1996.
- 3525 [tipsi] IEEE 1284.1, Transport-independent Printer System Interface (TIPSI).
- 3526 [URI-spec] Berners-Lee, T., Masinter, L., McCahill, M., "Uniform Resource Locators  
3527 (URL)", RFC 1738, December, 1994.

3528 [US-ASCII] Coded Character Set - 7-bit American Standard Code for Information  
3529 Interchange, ANSI X3.4-1986.  
3530 [UTF-8] F. Yergeau, "UTF-8, a transformation format of Unicode and ISO 10646", RFC  
3531 2044, October 1996.

3532 **8. Author's Addresses**

3533 Ron Bergman  
3534 Dataproducts Corp.  
3535 1757 Tapo Canyon Road  
3536 Simi Valley, CA 93063-3394  
3537

3538 Phone: 805-578-4421  
3539 Fax: 805-578-4001  
3540 Email: rbergman@dpc.com  
3541

3542  
3543 Tom Hastings  
3544 Xerox Corporation, ESAE-231  
3545 701 S. Aviation Blvd.  
3546 El Segundo, CA 90245  
3547  
3548 Phone: 310-333-6413  
3549 Fax: 310-333-5514  
3550 EMail: hastings@cp10.es.xerox.com  
3551

3552  
3553 Scott A. Isaacson  
3554 Novell, Inc.  
3555 122 E 1700 S  
3556 Provo, UT 84606  
3557  
3558 Phone: 801-861-7366  
3559 Fax: 801-861-4025  
3560 EMail: scott\_isaacson@novell.com  
3561

3562  
3563 Harry Lewis  
3564 IBM Corporation  
3565 6300 Diagonal Hwy  
3566 Boulder, CO 80301

3567

3568

Phone: (303) 924-5337

3569

Fax:

3570

Email: [harry1@us.ibm.com](mailto:harry1@us.ibm.com)

3571

3572

3573

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

3574

Mailing List: [jmp@pwg.org](mailto:jmp@pwg.org)

3575

3576

To learn how to subscribe, send email to: [jmp-request@pwg.org](mailto:jmp-request@pwg.org)

3577

3578

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

3579

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

3580

3581

Other Participants:

3582

Chuck Adams - Tektronix

3583

Jeff Barnett - IBM

3584

Keith Carter, IBM Corporation

3585

Jeff Copeland - QMS

3586

Andy Davidson - Tektronix

3587

Roger deBry - IBM

3588

Mabry Dozier - QMS

3589

Lee Ferrel - Canon

3590

Steve Gebert - IBM

3591

Robert Herriot - Sun Microsystems Inc.

3592

Shige Kanemitsu - Kyocera

3593

David Kellerman - Northlake Software

3594

Rick Landau - Digital

3595

Harry Lewis - IBM

3596

Pete Loya - HP

3597

Ray Lutz - Cognisys

3598

Jay Martin - Underscore

3599

Mike MacKay, Novell, Inc.

3600

Stan McConnell - Xerox

3601

Carl-Uno Manros, Xerox, Corp.

3602

Pat Nogay - IBM

3603

Bob Pentecost - HP

3604

Rob Rhoads - Intel

3605

David Roach - Unisys

3606

Hiroyuki Sato - Canon



3607 Bob Setterbo - Adobe  
3608 Gail Songer, EFI  
3609 Mike Timperman - Lexmark  
3610 Randy Turner - Sharp  
3611 William Wagner - Digital Products  
3612 Jim Walker - Dazel  
3613 Chris Wellens - Interworking Labs  
3614 Rob Whittle - Novell  
3615 Don Wright - Lexmark  
3616 Lloyd Young - Lexmark  
3617 Atsushi Yuki - Kyocera  
3618 Peter Zehler, Xerox, Corp.

3619 **9. INDEX**

3620 This index includes the textual conventions, the objects, and the attributes. Textual  
 3621 conventions all start with the prefix: "JM" and end with the suffix: "TC". Objects all  
 3622 starts with the prefix: "jm" followed by the group name. Attributes are identified with  
 3623 enums, and so start with any lower case letter and have no special prefix.

		3657	jmGeneralNewestActiveJobIndex .....	69	
3624	—C—	3658	jmGeneralNumberOfActiveJobs .....	69	
		3659	jmGeneralOldestActiveJobIndex .....	69	
3625	colorantConsumed .....	55	3660	jmJobIDJobIndex .....	73
3626	colorantRequested .....	55	3661	jmJobIDJobSetIndex .....	72
		3662	jmJobImpressionsCompleted .....	76	
3627	—D—	3663	jmJobImpressionsRequested .....	76	
		3664	jmJobIndex .....	74	
3628	deviceNameRequested .....	47	3665	jmJobKOctetsProcessed .....	75
3629	documentCopiesCompleted .....	52	3666	jmJobKOctetsRequested .....	75
3630	documentCopiesRequested .....	51	3667	jmJobOwner .....	77
3631	documentFormat .....	49	3668	JmJobServiceTypesTC .....	58
3632	documentFormatIndex .....	48	3669	JmJobSourcePlatformTypeTC .....	33
3633	documentName .....	48	3670	jmJobState .....	74
		3671	jmJobStateReasons1 .....	74	
3634	—F—	3672	JmJobStateReasons1TC .....	60	
		3673	JmJobStateReasons2TC .....	63	
3635	fileName .....	48	3674	JmJobStateReasons3TC .....	67
3636	finishing .....	50	3675	JmJobStateReasons4TC .....	67
3637	fullColorImpressionsCompleted .....	53	3676	JmJobStateTC .....	41
		3677	JmJobStringTC .....	32	
3638	—H—	3678	jmJobSubmissionID .....	72	
		3679	JmJobSubmissionTypeTC .....	38	
3639	highlightColorImpressionsCompleted .....	53	3680	JmMediumTypeTC .....	37
		3681	jmNumberOfInterveningJobs .....	75	
3640	—I—	3682	JmPrinterResolutionTC .....	36	
		3683	JmPrintQualityTC .....	35	
3641	impressionsCompletedCurrentCopy .....	53	3684	JmTimeStampTC .....	33
3642	impressionsInterpreted .....	52	3685	JmTonerEconomyTC .....	36
3643	impressionsSentToDevice .....	52	3686	JmUTF8StringTC .....	32
3644	impressionsSpooled .....	52	3687	jobAccountName .....	46
		3688	jobCodedCharSet .....	45	
3645	—J—	3689	jobComment .....	48	
		3690	jobCompletionTime .....	56	
3646	jmAttributeInstanceIndex .....	79	3691	jobCopiesCompleted .....	51
3647	jmAttributeTypeIndex .....	79	3692	jobCopiesRequested .....	51
3648	JmAttributeTypeTC .....	43	3693	jobHold .....	50
3649	jmAttributeValueAsInteger .....	79	3694	jobHoldUntil .....	50
3650	jmAttributeValueAsOctets .....	80	3695	jobKOctetsTransferred .....	52
3651	JmBooleanTC .....	36	3696	jobName .....	46
3652	JmFinishingTC .....	34	3697	jobOriginatingHost .....	47
3653	jmGeneralAttributePersistence .....	70	3698	jobPriority .....	49
3654	jmGeneralJobPersistence .....	70	3699	jobProcessAfterDateAndTime .....	49
3655	jmGeneralJobSetIndex .....	68	3700	jobProcessingCPUTime .....	56
3656	jmGeneralJobSetName .....	70	3701	jobServiceTypes .....	47

3702	jobSourceChannelIndex .....	47	3723	pagesRequested .....	53
3703	jobSourcePlatformType .....	47	3724	physicalDevice .....	48
3704	jobStartedBeingHeldTime .....	56	3725	printerResolutionRequested .....	51
3705	jobStartedProcessingTime .....	56	3726	printerResolutionUsed .....	51
3706	jobStateReasons2 .....	44	3727	printQualityRequested .....	50
3707	jobStateReasons3 .....	44	3728	printQualityUsed .....	50
3708	jobStateReasons4 .....	45	3729	processingMessage .....	45
3709	jobSubmissionTime .....	56			
3710	jobSubmissionToServerTime .....	56	3730	<b>—Q—</b>	
3711	jobURI .....	45			
			3731	queueNameRequested .....	48
3712	<b>—M—</b>				
			3732	<b>—S—</b>	
3713	mediumConsumed .....	55			
3714	mediumRequested .....	54	3733	serverAssignedJobName .....	46
			3734	sheetsCompleted .....	54
3715	<b>—N—</b>		3735	sheetsCompletedCurrentCopy .....	54
			3736	sheetsRequested .....	54
3716	numberOfDocuments .....	48	3737	sides .....	50
			3738	submittingApplicationName .....	47
3717	<b>—O—</b>		3739	submittingServerName .....	47
3718	other .....	44	3740	<b>—T—</b>	
3719	outputBin .....	50			
			3741	tonerDensityRequested .....	51
3720	<b>—P—</b>		3742	tonerDensityUsed .....	51
			3743	tonerEcomonyRequested .....	51
3721	pagesCompleted .....	53	3744	tonerEcomonyUsed .....	51
3722	pagesCompletedCurrentCopy .....	54			
3745					