

1 **Job Monitoring MIB, V0.89**

2 (This cover page is *not* part of the Internet-Draft
3 that is being forwarded to the IESG to be an Informational RFC)

4
5 From: Tom Hastings

6 Date: 12/12/97

7 Version: 0.89 (already numbered 1.0 in body, waiting for proof reading)

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

9 Status: Eleventh and Final draft MIB that incorporates the agreements reached at the
10 JMP Meeting, on 12/5/97 in L.A. on issues in V0.87 which was released after the 10/31
11 meeting. The changes include:

- 12 1. use the new PWG OIDs without the standard arc.
- 13 2. make the document a PWG draft standard that will be sent as an Internet-
14 Draft that will become an IETF Informational RFC, including changing the
15 IANA Considerations section
- 16 3. add natural language support like IPP
- 17 4. fix the issues with monitoring collated/uncollated implementations
- 18 5. fix impressions completed,
- 19 6. allows multiple Job Submission Id entries to point to the same jmJobIndex
20 entry
- 21 7. and add 3 new Job Submission Ids

22 See the change history in the separate file: changes.doc .pdf.

23 We agreed that the MIB specification is finished except for any editorial comments that
24 people may have. See the separate issues.doc and .pdf file.

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

28 The MIB has been greatly simplified so that now there are only 18 objects in the MIB.
29 There are 73 attributes.

31 INTERNET-DRAFT

32

33

34

35

36

37

38

39

R. Bergman
Dataproducts Corp.
T. Hastings
Xerox Corporation
S. Isaacson
Novell, Inc.
H. Lewis
IBM Corp.
December 12, 1997

40

Job Monitoring MIB - V1

41

<draft-ietf-printmib-job-monitor-07.txt>

42

Status of this Memo

43

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

44

45

46

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

47

48

49

50

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

51

52

53

54

This Internet-Draft expires on June 12, 1997.

55

Abstract

56

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

57

58

59

60

61

62

63

64

65

66

67

68

TABLE OF CONTENTS

69	1. INTRODUCTION.....	10
70	1.1 Types of Information in the MIB	10
71	1.2 Types of Job Monitoring Applications	11
72	2. TERMINOLOGY AND JOB MODEL.....	12
73	2.1 System Configurations for the Job Monitoring MIB	15
74	2.1.1 Configuration 1 - client-printer	16
75	2.1.2 Configuration 2 - client-server-printer - agent in the server	16
76	2.1.3 Configuration 3 - client-server-printer - client monitors printer agent and server	18
77	3. MANAGED OBJECT USAGE.....	19
78	3.1 Conformance Considerations	19
79	3.1.1 Conformance Terminology	20
80	3.1.2 Agent Conformance Requirements.....	20
81	3.1.2.1 MIB II System Group objects.....	20
82	3.1.2.2 MIB II Interface Group objects	20
83	3.1.2.3 Printer MIB objects	21
84	3.1.3 Job Monitoring Application Conformance Requirements.....	21
85	3.2 The Job Tables and the Oldest Active and Newest Active Indexes	21
86	3.3 The Attribute Mechanism.....	23
87	3.3.1 Conformance of Attribute Implementation.....	24
88	3.3.2 Useful, 'Unknown', and 'Other' Values for Objects and Attributes.....	24
89	3.3.3 Data Sub-types and Attribute Naming Conventions.....	25
90	3.3.4 Single-Value (Row) Versus Multi-Value (MULTI-ROW) Attributes.....	26
91	3.3.5 Requested Objects and Attributes	26
92	3.3.6 Consumption Attributes	26
93	3.3.7 Index Value Attributes	26
94	3.4 Monitoring Job Progress	27
95	3.5 Job Identification.....	30
96	3.6 Internationalization Considerations	31
97	3.6.1 Text generated by the server or device.....	31
98	3.6.2 Text supplied by the job submitter	32
99	3.6.3 'DateAndTime' for representing the date and time.....	33
100	3.7 IANA and PWG Registration Considerations.....	33

101	3.7.1	PWG Registration of enums.....	33
102	3.7.1.1	Type 1 enumerations	34
103	3.7.1.2	Type 2 enumerations	34
104	3.7.1.3	Type 3 enumeration.....	34
105	3.7.2	PWG Registration of type 2 bit values.....	34
106	3.7.3	PWG Registration of Job Submission Id Formats.....	35
107	3.7.4	PWG Registration of MIME types/sub-types for document-formats.....	35
108	3.8	Security Considerations	35
109	3.8.1	Read-Write objects.....	35
110	3.8.2	Read-Only Objects In Other User's Jobs.....	35
111	3.9	Notifications	36
112	4.	MIB SPECIFICATION.....	36
113		Textual conventions for this MIB module.....	37
114		JmUTF8StringTC.....	38
115		JmJobStringTC	38
116		JmTimeStampTC	38
117		JmJobSourcePlatformTypeTC	39
118		JmFinishingTC	39
119		JmPrintQualityTC	41
120		JmPrinterResolutionTC.....	41
121		JmTonerEconomyTC.....	41
122		JmBooleanTC	42
123		JmMediumTypeTC	42
124		JmJobCollationTypeTC.....	44
125		JmJobStateTC	47
126		JmAttributeTypeTC.....	50
127		other (Int32(-2..) and/or Octets63).....	50
128		Job State attributes.....	50
129		jobStateReasons2 (JmJobStateReasons2TC).....	51
130		jobStateReasons3 (JmJobStateReasons3TC).....	51
131		jobStateReasons4 (JmJobStateReasons4TC).....	51
132		processingMessage (UTF8String63).....	51
133		processingMessageNaturalLanguageTag (Octets63).....	51
134		jobCodedCharSet (CodedCharSet).....	52
135		jobNaturalLanguageTag (Octets63).....	52
136		Job Identification attributes	52
137		jobURI (Octets(0..63)).....	52
138		jobAccountName (Octets63).....	53
139		serverAssignedJobName (JobString63).....	53
140		jobName (JobString63).....	53
141		jobServiceTypes (JmJobServiceTypesTC).....	54
142		jobSourceChannelIndex (Int32(0..)).....	54
143		jobSourcePlatformType (JmJobSourcePlatformTypeTC).....	54
144		submittingServerName (JobString63).....	54
145		submittingApplicationName (JobString63).....	54

146	jobOriginatingHost (JobString63)	54
147	deviceNameRequested (JobString63)	54
148	queueNameRequested (JobString63)	55
149	physicalDevice (hrDeviceIndex and/or UTF8String63)	55
150	numberOfDocuments (Int32(-2..))	55
151	fileName (JobString63)	55
152	documentName (JobString63)	55
153	jobComment (JobString63)	55
154	documentFormatIndex (Int32(0..))	56
155	documentFormat (PrtInterpreterLangFamilyTC and/or Octets63)	56
156	Job Parameter attributes	56
157	jobPriority (Int32(-2..100))	56
158	jobProcessAfterDateAndTime (DateAndTime)	57
159	jobHold (JmBooleanTC)	57
160	jobHoldUntil (JobString63)	57
161	outputBin (Int32(0..) and/or JobString63)	57
162	sides (Int32(-2..2))	57
163	finishing (JmFinishingTC)	57
164	Image Quality attributes (requested and used)	57
165	printQualityRequested (JmPrintQualityTC)	58
166	printQualityUsed (JmPrintQualityTC)	58
167	printerResolutionRequested (JmPrinterResolutionTC)	58
168	printerResolutionUsed (JmPrinterResolutionTC)	58
169	tonerEcomonyRequested (JmTonerEconomyTC)	58
170	tonerEcomonyUsed (JmTonerEconomyTC)	58
171	tonerDensityRequested (Int32(-2..100))	58
172	tonerDensityUsed (Int32(-2..100))	58
173	Job Progress attributes (requested and consumed)	58
174	jobCopiesRequested (Int32(-2..))	58
175	jobCopiesCompleted (Int32(-2..))	59
176	documentCopiesRequested (Int32(-2..))	59
177	documentCopiesCompleted (Int32(-2..))	59
178	jobKOctetsTransferred (Int32(-2..))	59
179	sheetCompletedCopyNumber (Int32(-2..))4	59
180	sheetCompletedDocumentNumber (Int32(-2..))4	59
181	jobCollationType JmJobCollationTypeTC)	60
182	Impression attributes (requested and consumed)	60
183	impressionsSpooled (Int32(-2..))	60
184	impressionsSentToDevice (Int32(-2..))	60
185	impressionsInterpreted (Int32(-2..))	60
186	impressionsCompletedCurrentCopy (Int32(-2..))	60
187	fullColorImpressionsCompleted (Int32(-2..))	60
188	highlightColorImpressionsCompleted (Int32(-2..))	60
189	Page attributes (requested and consumed)	61
190	pagesRequested (Int32(-2..))	61
191	pagesCompleted (Int32(-2..))	61
192	pagesCompletedCurrentCopy (Int32(-2..))	61
193	Sheet attributes (requested and consumed)	61
194	sheetsRequested (Int32(-2..))	62

195	sheetsCompleted (Int32(-2..)).....	62
196	sheetsCompletedCurrentCopy (Int32(-2..)).....	62
197	Resource attributes (requested and consumed).....	62
198	mediumRequested (JmMediumTypeTC and/or JobString63).....	62
199	mediumConsumed (Int32(-2..) and/or JobString63).....	62
200	colorantRequested (Int32(-2..) and/or JobString63).....	63
201	colorantConsumed (Int32(-2..) and/or JobString63).....	63
202	Time attributes (set by server or device).....	63
203	jobSubmissionToServerTime (JmTimeStampTC and/or DateAndTime).....	64
204	jobSubmissionTime (JmTimeStampTC and/or DateAndTime).....	64
205	jobStartedBeingHeldTime (JmTimeStampTC).....	64
206	jobStartedProcessingTime (JmTimeStampTC and/or DateAndTime).....	64
207	jobCompletionTime (JmTimeStampTC and/or DateAndTime).....	64
208	jobProcessingCPUTime (Int32(-2..)).....	64
209	JmJobServiceTypesTC.....	66
210	JmJobStateReasons1TC.....	68
211	JmJobStateReasons2TC.....	71
212	JmJobStateReasons3TC.....	74
213	JmJobStateReasons4TC.....	75
214	The General Group (MANDATORY).....	76
215	jmGeneralJobSetIndex (Int32(1..32767)).....	76
216	jmGeneralNumberOfActiveJobs (Int32(0..)).....	77
217	jmGeneralOldestActiveJobIndex (Int32(0..)).....	77
218	jmGeneralNewestActiveJobIndex (Int32(0..)).....	77
219	jmGeneralJobPersistence (Int32(15..)).....	78
220	jmGeneralAttributePersistence (Int32(15..)).....	78
221	jmGeneralJobSetName (UTF8String63).....	78
222	The Job ID Group (MANDATORY).....	79
223	jmJobSubmissionID (OCTET STRING(SIZE(48))).....	80
224	jmJobIDJobSetIndex (Int32(0..32767)).....	80
225	jmJobIDJobIndex (Int32(0..)).....	81
226	The Job Group (MANDATORY).....	81
227	jmJobIndex (Int32(1..)).....	82
228	jmJobState (JmJobStateTC).....	82
229	jmJobStateReasons1 (JmJobStateReasons1TC).....	83
230	jmNumberOfInterveningJobs (Int32(-2..)).....	83
231	jmJobKOctetsPerCopyRequested (Int32(-2..)).....	83
232	jmJobKOctetsProcessed (Int32(-2..)).....	84
233	jmJobImpressionsPerCopyRequested (Int32(-2..)).....	84
234	jmJobImpressionsCompleted (Int32(-2..)).....	85
235	jmJobOwner (JobString63).....	85
236	The Attribute Group (MANDATORY).....	86
237	jmAttributeTypeIndex (JmAttributeTypeTC).....	87
238	jmAttributeInstanceIndex (Int32(1..32767)).....	87
239	jmAttributeValueAsInteger (Int32(-2..)).....	88
240	jmAttributeValueAsOctets (Octets63).....	89

241 **5. APPENDIX A - IMPLEMENTING THE JOB LIFE CYCLE..... 92**

242 **6. APPENDIX B - SUPPORT OF JOB SUBMISSION PROTOCOLS 92**

243 **7. REFERENCES..... 93**

244 **8. AUTHOR'S ADDRESSES 94**

245 **9. INDEX..... 98**

246

247

Job Monitoring MIB

248 1. Introduction

249 This specification defines an official Printer Working Group (PWG) [PWG] standard
250 SNMP MIB for the monitoring of jobs on network printers. This specification is being
251 published as an IETF Information Document for the convenience of the Internet
252 community. In consultation with the IETF Application Area Directors, we concluded that
253 it properly belongs as an Information document, because this MIB monitors a service
254 node on the network, rather than a network node proper.

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

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

275 1.1 Types of Information in the MIB

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

278 User:

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

282 Provide the ability to identify the current status of the user's job (user queries).
283 Provide a timely indication that the job has completed and where it can be found.
284 Provide error and diagnostic information for jobs that did not successfully
285 complete.

286 Operator:

287 Provide a presentation of the state of all the jobs in the print system.
288 Provide the ability to identify the user that submitted the print job.
289 Provide the ability to identify the resources required by each job.
290 Provide the ability to define which physical printers are candidates for the print
291 job.
292 Provide some idea of how long each job will take. However, exact estimates of
293 time to process a job is not being attempted. Instead, objects are included that
294 allow the operator to be able to make gross estimates.

295 Capacity Planner:

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

298 Accountant:

299 Provide information to allow the creation of a record of resources consumed and
300 printer usage data for charging users or groups for resources consumed.
301 Provide information to allow the prediction of consumable usage and resource
302 need.

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

308 1.2 Types of Job Monitoring Applications

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

- 310 1. Monitor a single job starting when the job is submitted and ending a defined
311 period after the job completes. The Job Submission ID table provides the
312 map to find the specific job to be monitored.
- 313 2. Monitor all 'active' jobs in a queue, which this specification generalizes to a
314 "job set". End users may use such a program when selecting a least busy

315 printer, so the MIB is designed for such a program to start up quickly and find
316 the information needed quickly without having to read all (completed) jobs in
317 order to find the active jobs. System operators may also use such a program,
318 in which case it would be running for a long period of time and may also be
319 interested in the jobs that have completed. Finally such a program may be
320 used to provide an enhanced console and logging capability.

321 3. Collect resource usage for accounting or system utilization purposes that copy
322 the completed job statistics to an accounting system. It is recognized that
323 depending on accounting programs to copy MIB data during the job-retention
324 period is somewhat unreliable, since the accounting program may not be
325 running (or may have crashed). Such a program is also expected to keep a
326 shadow copy of the entire Job **Attribute** table including **completed,**
327 **canceled, and aborted** jobs which the program updates on each polling
328 cycle. Such a program polls at the rate of the persistence of the **Attribute**
329 table. The design is not optimized to help such an application determine
330 which jobs are **completed, canceled, or aborted**. Instead, the application
331 SHALL query each job that the application's shadow copy shows was not
332 **complete, canceled, or aborted** at the previous poll cycle to see if it is now
333 **complete or canceled**, plus any new jobs that have been submitted.

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

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

344 2. Terminology and Job Model

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

347 NOTE - Existing systems use conflicting terms, so these terms are drawn from the ISO
348 10175 Document Printing Application (DPA) standard[iso-dpa]. For example,
349 PostScript systems use the term *session* for what is called a *job* in this specification
350 and the term *job* to mean what is called a *document* in this specification.

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

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

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

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

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

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

376 Document Instruction: An instruction specifying how to process the document.
377 Document instructions MAY be passed in the job submission protocol separate from the
378 actual document data, or MAY be embedded in the document data or a combination,
379 depending on the job submission protocol and implementation.

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

381 Impression: For a print job, an impression is the passage of the entire side of a sheet by
382 the marker, whether or not any marks are made and independent of the number of passes
383 that the side makes past the marker. Thus a four pass color process counts as a single
384 impression, as does highlight color. Impression counters count all kinds: monochrome,
385 highlight color, and full process color, while full color counters only count full color
386 impressions, and high light color counters only count high light color impressions.

387 One-sided processing involves one impression per sheet. Two-sided processing involves
388 two impressions per sheet. If a two-sided document has an odd number of pages, the last
389 sheet still counts as two impressions, if that sheet makes two passes through the marker
390 or the marker marks on both sides of a sheet in a single pass. Two-up printing is the

391 placement of two logical pages on one side of a sheet and so is still a single impression.
392 See "page" and "sheet".

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

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

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

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

404 Job Instruction: An instruction specifying how, when, or where the job is to be
405 processed. Job instructions MAY be passed in the job submission protocol or MAY be
406 embedded in the document data or a combination depending on the job submission
407 protocol and implementation.

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

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

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

422 Monitor: Short for Job Monitoring Application.

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

426 Proxy: An agent that acts as a concentrator for one or more other agents by accepting
427 SNMP operations on the behalf of one or more other agents, forwarding them on to those

- 428 other agents, gathering responses from those other agents and returning them to the
429 original requesting monitor.
- 430 Queuing: The act of a *device* or *server* of ordering (queuing) the jobs for the purposes of
431 scheduling the jobs to be processed.
- 432 Printer: A *device* that puts marks on media.
- 433 Server: A network entity that accepts jobs from clients and in turn submits the jobs to
434 *printers* and other *devices* that may be directly connected to the server via a serial or
435 parallel port or may be on the network. A server MAY be a printer *supervisor* control
436 program, or a print *spooler*.
- 437 Sheet: A sheet is a single instance of a medium, whether printing on one or both sides of
438 the medium. See "impression" and "page".
- 439 SNMP Information Object: A name, value-pair that specifies an action, a status, or a
440 condition in an SNMP MIB. Objects are identified in SNMP by an OBJECT
441 IDENTIFIER.
- 442 Spooler: A server that accepts jobs, spools the data, and decides when and on which
443 printer to print the job. A spooler is a client to a printer or a printer supervisor, depending
444 on implementation.
- 445 Spooling: The act of a *device* or *server* of (1) accepting jobs and (2) writing the job's
446 attributes and document data on to secondary storage.
- 447 Stacked: When a media sheet is placed in an output bin of a device.
- 448 Supervisor: A server that contains a control program that controls a printer or other
449 device. A supervisor is a client to the printer or other device.
- 450 System Operator: A user that uses a monitor to monitor the system and carries out tasks
451 to keep the system running.
- 452 System Administrator: A user that specifies policy for the system.
- 453 Two-up: The placement of two pages on one side of a sheet so that each side or
454 impressions counts as two pages. See "page" and "sheet".
- 455 User: A person that uses a client or a monitor. See "end user".

456 2.1 System Configurations for the Job Monitoring MIB

457 This section enumerates the three configurations in which the Job Monitoring MIB is
458 intended to be used. To simplify the pictures, the *devices* are shown as *printers*. See
459 section 0 entitled "Types of Information in the MIB".

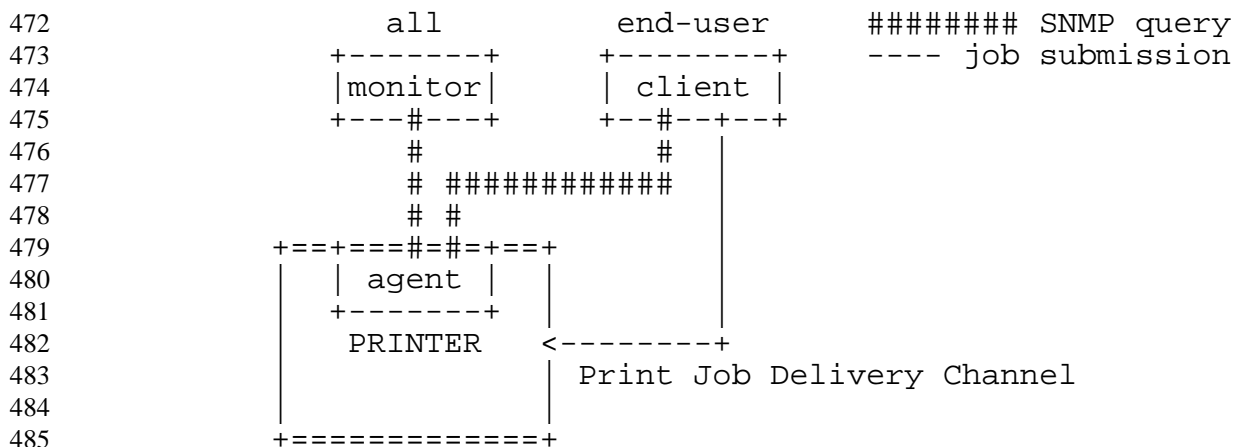
460 The diagram in the Printer MIB[print-mib] entitled: "One Printer's View of the Network"
 461 is assumed for this MIB as well. Please refer to that diagram to aid in understanding the
 462 following system configurations.

463 **2.1.1 Configuration 1 - client-printer**

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

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

471



486 **Figure 0-1 - Configuration 1 - client-printer - agent in the printer**

487 The Job Monitoring MIB is designed to support the following relationships (not shown in
 488 Figure 0-1):

- 489 1. Multiple **clients** MAY submit jobs to a **printer**.
- 490 2. Multiple **clients** MAY monitor a **printer**.
- 491 3. Multiple **monitors** MAY monitor a **printer**.
- 492 4. A **client** MAY submit jobs to multiple **printers**.
- 493 5. A **monitor** MAY monitor multiple **printers**.

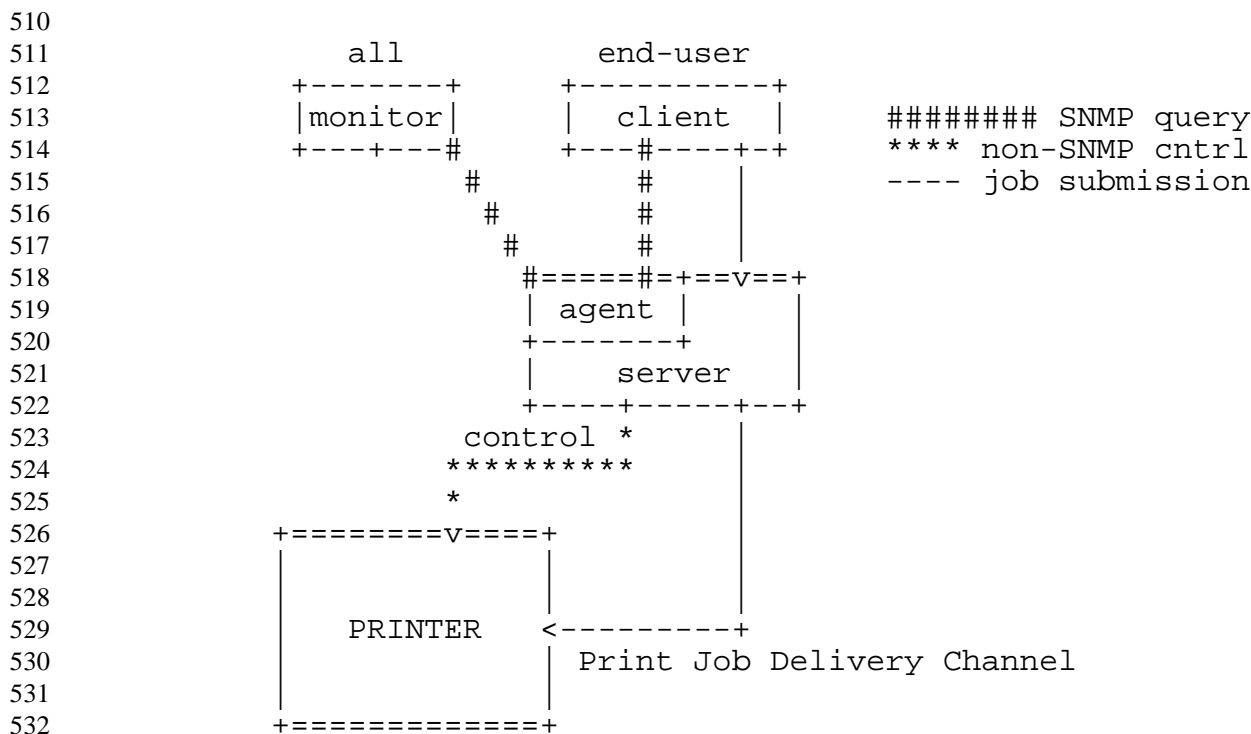
494 **2.1.2 Configuration 2 - client-server-printer - agent in the server**

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

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

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

501 There is no SNMP Job Monitoring MIB agent in the **printer** in configuration 2, at least
 502 that the client or monitor are aware. In this configuration, the agent SHALL return the
 503 current values of the objects in the Job Monitoring MIB both for jobs the server keeps
 504 and jobs that the server has submitted to the **printer**. The Job Monitoring MIB agent
 505 SHALL obtain the required information from the **printer** by a method that is beyond the
 506 scope of this document. The agent in the **server** SHALL keep the job in the Job
 507 Monitoring MIB in the server as long as the job is in the **printer**, plus a defined time
 508 period after the job enters the **completed** state in which accounting programs can copy
 509 out the accounting data from the Job Monitoring MIB.



533 **Figure 0-2 - Configuration 2 - client-server-printer - agent in the server**

534 The Job Monitoring MIB is designed to support the following relationships (not shown in
 535 Figure 0-2):

- 536 1. Multiple **clients** MAY submit jobs to a **server**.
- 537 2. Multiple **clients** MAY monitor a **server**.
- 538 3. Multiple **monitors** MAY monitor a **server**.
- 539 4. A **client** MAY submit jobs to multiple **servers**.
- 540 5. A **monitor** MAY monitor multiple **servers**.

- 541 6. Multiple **servers** MAY submit jobs to a **printer**.
542 7. Multiple **servers** MAY control a **printer**.

543 **2.1.3 Configuration 3 - client-server-printer - client monitors printer agent and**
544 **server**

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

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

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

557 In configuration 3, the agent (in the **printer**) SHALL keep the values of the objects in the
558 Job Monitoring MIB that the agent implements updated for a job that the server has
559 submitted to the printer. The agent SHALL obtain information about the jobs submitted
560 to the printer from the server (either in the job submission protocol, in the document data,
561 or by direct query of the server), in order to populate some of the objects the Job
562 Monitoring MIB in the printer. The agent in the printer SHALL keep the job in the Job
563 Monitoring MIB as long as the job is in the Printer, and longer in order to implement the
564 **completed** state in which monitoring programs can copy out the accounting data from the
565 Job Monitoring MIB.

606 3.1.1 Conformance Terminology

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

- 609 • "SHALL": indicates an action that the subject of the sentence must implement in
610 order to claim conformance to this specification
- 611 • "MAY": indicates an action that the subject of the sentence does not have to
612 implement in order to claim conformance to this specification, in other words that
613 action is an implementation option
- 614 • "NEED NOT": indicates an action that the subject of the sentence does not have to
615 implement in order to claim conformance to this specification. The verb "NEED
616 NOT" is used instead of "may not", since "may not" sounds like a prohibition.
- 617 • "SHOULD": indicates an action that is recommended for the subject of the
618 sentence to implement, but is not required, in order to claim conformance to this
619 specification.

620 3.1.2 Agent Conformance Requirements

621 A conforming agent:

- 622 1. SHALL implement *all* MANDATORY groups in this specification.
- 623 2. SHALL implement any attributes if (1) the server or device supports the
624 functionality represented by the attribute and (2) the information is available
625 to the agent.
- 626 3. SHOULD implement both forms of an attribute if it implements an attribute
627 that permits a choice of INTEGER and OCTET STRING forms, since
628 implementing both forms may help management applications by giving them
629 a choice of representations, since the representation are equivalent. See the
630 **JmAttributeTypeTC** textual-convention.

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

633 3.1.2.1 MIB II System Group objects

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

636 3.1.2.2 MIB II Interface Group objects

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

639 3.1.2.3 Printer MIB objects

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

646 **3.1.3 Job Monitoring Application Conformance Requirements**

647 A conforming job monitoring application:

- 648 1. SHALL accept the full syntactic range for all objects in all MANDATORY
649 groups and all MANDATORY attributes that are required to be implemented
650 by an agent according to Section 0 and SHALL either present them to the user
651 or ignore them.
- 652 2. SHALL accept the full syntactic range for *all* attributes, including enum and
653 bit values specified in this specification and additional ones that may be
654 registered with the PWG and SHALL either present them to the user or ignore
655 them. In particular, a conforming job monitoring application SHALL not
656 malfunction when receiving any standard or registered enum or bit values.
657 See Section 0 entitled "IANA and PWG Registration Considerations".
- 658 3. SHALL NOT fail when operating with agents that materialize attributes *after*
659 the job has been submitted, as opposed to when the job is submitted.
- 660 4. SHALL, if it supports a time attribute, accept either form of the time attribute,
661 since agents are free to implement either time form.

662 **3.2 The Job Tables and the Oldest Active and Newest Active Indexes**

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

- 665 1. **jmGeneralJobSetIndex** - which job set
- 666 2. **jmJobIndex** - which job in the job set

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

- 669 1. **jmGeneralOldestActiveJobIndex** - the index of the active job that has been
670 in the tables the longest.
- 671 2. **jmGeneralNewestActiveJobIndex** - the index of the active job that has been
672 most recently added to the tables.

673 The agent SHALL assign the next incremental value of **jmJobIndex** to the job, when a
674 new job is accepted by the server or device to which the agent is providing access. If the
675 incremented value of **jmJobIndex** would exceed the implementation-defined maximum

676 value for **jmJobIndex**, the agent SHALL 'wrap' back to 1. An agent uses the resulting
677 value of **jmJobIndex** for storing information in the **jmJobTable** and the
678 **jmAttributeTable** about the job.

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

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

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

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

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

709 Whenever a job transitions from one of the 'inactive' job states to one of the 'active' job
710 states (from **pendingHeld** to **pending** or **processing**), the agent SHALL update the value
711 of either the **jmGeneralOldestActiveJobIndex** or the
712 **jmGeneralNewestActiveJobIndex** objects, or both, if the job's **jmJobIndex** value is

713 outside the range between **jmGeneralOldestActiveJobIndex** and
714 **jmGeneralNewestActiveJobIndex**.

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

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

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

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

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

733 3.3 The Attribute Mechanism

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

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

- 741 1. **jmGeneralJobSetIndex** - which job set
- 742 2. **jmJobIndex** - which job in the job set
- 743 3. **jmAttributeTypeIndex** - which attribute
- 744 4. **jmAttributeInstanceIndex** - which attribute instance for those attributes that
745 can have multiple values per job.

746 Some attributes represent information about a job, such as a file-name, a document-name,
747 a submission-time or a completion time. Other attributes represent resources required,

748 e.g., a medium or a colorant, etc. to process the job before the job starts processing OR to
749 indicate the amount of the resource consumed during and after processing, e.g., pages
750 completed or impressions completed. If both a required and a consumed value of a
751 resource is needed, this specification assigns two separate attribute enums in the textual
752 convention.

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

760 3.3.1 Conformance of Attribute Implementation

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

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

769 3.3.2 Useful, 'Unknown', and 'Other' Values for Objects and Attributes

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

774 SNMP requires that if an object cannot be implemented because its values cannot be
775 accessed, then a compliant agent SHALL return an SNMP error in SNMPv1 or an
776 exception value in SNMPv2. However, this MIB has been designed so that 'all' objects
777 can and SHALL be implemented by an agent, so that neither the SNMPv1 error nor the
778 SNMPv2 exception value SHALL be generated by the agent. This MIB has also been
779 designed so that when an agent materializes an attribute, the agent SHALL materialize a
780 row consisting of both the **jmAttributeValueAsInteger** and **jmAttributeValueAsOctets**
781 objects.

782 In general, values for objects and attributes have been chosen so that a management
783 application will be able to determine whether a 'useful', 'unknown', or 'other' value is

784 available. When a useful value is not available for an object that agent SHALL return a
 785 zero-length string for octet strings, the value **'unknown(2)'** for enums, a **'0'** value for an
 786 object that represents an index in another table, and a value **'-2'** for counting integers.

787 Since each attribute is represented by a row consisting of both the
 788 **jmAttributeValueAsInteger** and **jmAttributeValueAsOctets** MANDATORY objects,
 789 SNMP requires that the agent SHALL always create an attribute row with both objects
 790 specified. However, for most attributes the agent SHALL return a "useful" value for one
 791 of the objects and SHALL return the 'other' value for the other object. For integer only
 792 attributes, the agent SHALL always return a zero-length string value for the
 793 **jmAttributeValueAsOctets** object. For octet string only attributes, the agent SHALL
 794 always return a **'-1'** value for the **jmAttributeValueAsInteger** object.

795 3.3.3 Data Sub-types and Attribute Naming Conventions

796 Many attributes are sub-typed to give a more specific data type than **Integer32** or
 797 **OCTET STRING**. The data sub-type of each attribute is indicated on the first line(s) of
 798 the description. Some attributes have several different data sub-type representations.
 799 When an attribute has both an **Integer32** data sub-type and an **OCTET STRING** data
 800 sub-type, the attribute can be represented in a single row in the **jmAttributeTable**. In
 801 this case, the data sub-type name is not included as the last part of the name of the
 802 attribute, e.g., **documentFormat(38)** which is both an enum and/or a name. When the
 803 data sub-types cannot be represented by a single row in the **jmAttributeTable**, each such
 804 representation is considered a separate attribute and is assigned a separate name and enum
 805 value. For these attributes, the name of the data sub-type is the last part of the name of
 806 the attribute: **Name**, **Index**, **DateAndTime**, **TimeStamp**, etc. For example,
 807 **documentFormatIndex(37)** is an index.

808 NOTE: The Table of Contents also lists the data sub-type and/or data sub-types of each
 809 attribute, using the textual-convention name when such is defined. The following
 810 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.

811 3.3.4 Single-Value (Row) Versus Multi-Value (MULTI-ROW) Attributes

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

820 NOTE - Duplicates are allowed for 'extensive' 'MULTI-ROW' attributes, such as
821 **fileName(34)** or **documentName(35)** which are specified to be 'per-document' attributes,
822 but are *not* allowed for 'intensive' 'MULTI-ROW' attributes, such as
823 **mediumConsumed(171)** and **documentFormat(38)** which are specified to be 'per-job'
824 attributes.

825 3.3.5 Requested Objects and Attributes

826 A number of objects and attributes record requirements for the job. Such object and
827 attribute names end with the word '**Requested**'. In the interests of brevity, the phrase
828 'requested' SHALL mean: (1) requested by the client (or intervening server) in the job
829 submission protocol and MAY also mean (2) embedded in the submitted document data,
830 and/or (3) defaulted by the recipient device or server with the same semantics as if the
831 requester had supplied, depending on implementation. Also if a value is supplied by the
832 job submission client, and the server/device determines a better value, through processing
833 or other means, the agent MAY return that better value for such object and attribute.

834 3.3.6 Consumption Attributes

835 A number of objects and attributes record consumption. Such attribute names end with
836 the word '**Completed**' or '**Consumed**'. If the job has not yet consumed what that
837 resource is metering, the agent either: (1) SHALL return the value **0** or (2) SHALL *not*
838 add this attribute to the **jmAttributeTable** until the consumption begins. In the interests
839 of brevity, the semantics for **0** is specified once here and is *not* repeated for each
840 consumption attribute specification and a DEFVAL of 0 is indicated.

841 3.3.7 Index Value Attributes

842 A number of attributes are indexes in other tables. Such attribute names end with the
843 word '**Index**'. If the agent has not (yet) assigned an index value for a particular index
844 attribute for a job, the agent SHALL either: (1) return the value **0** or (2) *not* add this
845 attribute to the **jmAttributeTable** until the index value is assigned. In the interests of

846 brevity, the semantics for 0 is specified once here and is *not* repeated for each index
847 attribute specification and a DEFVAL of 0 is indicated.

848 3.4 Monitoring Job Progress

849 There are a number of objects and attributes for monitoring the progress of a job. These
850 objects and attributes count the number of K octets, impressions, sheets, and pages
851 requested or completed. For impressions and sheets, "completed" SHALL mean stacked,
852 unless the implementation is unable to detect when each sheet is stacked, in which case
853 stacked is approximated when processing of each sheet completes. There are objects and
854 attributes for the overall job and for the current copy of the document currently being
855 stacked. For the latter, the rate at which the various objects and attributes count depends
856 on the sheet and document collation of the job.

857 Job Collation included sheet collation and document collation. Sheet collation is defined
858 to be the ordering of sheets within a document copy. Document collation is defined to be
859 ordering of document copies within a multi-document job. There are three types of job
860 collation (see terminology definitions in Section 0):

861 1. Uncollated Sheets - No collation of the sheets within each document copy, i.e.,
862 each sheet of a document that is to produce multiple copies is replicated
863 before the next sheet in the document is processed and stacked. If the device
864 has an output bin collator, uncollated sheets may actually produce collated
865 sheets as far as the user is concerned (in the output bins). However, when the
866 job collation is 'uncollated sheets', job progress is indistinguishable to a
867 monitoring application between a device that has an output bin collator and
868 one that does not.

869 2. Collated Documents - Collation of the sheets within each document copy is
870 performed within the printing device by making multiple passes over either
871 the source or an intermediate representation of the document. In addition,
872 when there are multiple documents per job, the i'th copy of each document is
873 stacked before the j'th copy of each document, i.e., the documents are collated
874 within each job copy. For example, if a job is submitted with documents, A
875 and B, the job is made available to the end user as: A, B, A, B, Collated
876 Document correspond to the IPP [ipp-model] 'separate-documents-collated-
877 copies' value of the "multiple-document-handling" attribute.
878

879 If **jobCopiesRequested** or **documentCopiesRequested** = 1, then
880 **jobCollationType** is defined as 4.

881 3. Uncollated Documents - Collation of the sheets within each document copy is
882 performed within the printing device by making multiple passes over either
883 the source or an intermediate representation of the document. In addition,

884 when there are multiple documents per job, all copies of the first document in
 885 the job are stacked before the any copied of the next document in the job, i.e.,
 886 the documents are uncollated within the job. For example, if a job is
 887 submitted with documents, A and B, the job is mad available to the end user
 888 as: A, A, ..., B, B, Uncollated Documents correspond to the IPP [ipp-
 889 model] 'separate-documents-uncollated-copies' value of the "multiple-
 890 document-handling" attribute.

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

- 893 1. **jmJobImpressionsCompleted** - counts the total number of impressions
 894 stacked for the job
- 895 2. **impressionsCompletedCurrentCopy** - counts the number of impressions
 896 stacked for the current document copy
- 897 3. **sheetCompletedCopyNumber** - identifies the number of the copy for the
 898 current document being stacked where the first copy is 1.
- 899 4. **sheetCompletedDocumentNumber** - identifies the current document within
 900 the job that is being stacked where the first document in a job is 1. NOTE:
 901 this attribute SHOULD NOT be implemented for implementations that only
 902 support one document per job.

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

906 Job Collation Type = Uncollated Sheets

907

jmJobImpressions Completed	impressionsCompleted CurrentCopy	sheetCompleted CopyNumber	sheetCompleted DocumentNumber
0	0	0	0
1	1	1	1
2	1	2	1
3	1	3	1
4	2	1	1
5	2	2	1
6	2	3	1
7	3	1	1
8	3	2	1
9	3	3	1
10	1	1	2
11	1	2	2
12	1	3	2

13	2	1	2
14	2	2	2
15	2	3	2
16	3	1	2
17	3	2	2
18	3	3	2

908

909 Job Collation Type = Collated Documents

910

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

911

912 Job Collation Type = Uncollated Documents

913

jmJobImpressions Completed	impressionsCompleted CurrentCopy	sheetCompleted CopyNumber	sheetCompleted DocumentNumber
0	0	0	0
1	1	1	1
2	2	1	1
3	3	1	1
4	1	2	1
5	2	2	1
6	3	2	1
7	1	3	1

8	2	3	1
9	3	3	1
10	1	1	2
11	2	1	2
12	3	1	2
13	1	2	2
14	2	2	2
15	3	2	2
16	1	3	2
17	2	3	2
18	3	3	2

914

915 **3.5 Job Identification**

916 There are a number of attributes that permit a user, operator or system administrator to
 917 identify jobs of interest, such as **jobURI**, **jobName**, **jobOriginatingHost**, etc. In
 918 addition, there is a **jmJobSubmissionID** object that is a text string table index. Being a
 919 table index allows a monitoring application to quickly locate and identify a particular job
 920 of interest that was submitted from a particular client by the user invoking the monitoring
 921 application without having to scan the entire job table. The Job Monitoring MIB needs to
 922 provide for identification of the job at both sides of the job submission process. The
 923 primary identification point is the client side. The **jmJobSubmissionID** allows the
 924 monitoring application to identify the job of interest from all the jobs currently "known"
 925 by the server or device. The value of **jmJobSubmissionID** can be assigned by either the
 926 client's local system or a downstream server or device. The point of assignment depends
 927 on the job submission protocol in use.

928 The server/device-side identifier, called the **jmJobIndex** object, SHALL be assigned by
 929 the SNMP Job Monitoring MIB agent when the server or device accepts the jobs from
 930 submitting clients. The **jmJobIndex** object allows the interested party to obtain all
 931 objects desired that relate to a particular job. See Section 0, entitled 'The Job Tables and
 932 the Oldest Active and Newest Active Indexes' for the specification of how the agent
 933 SHALL assign the **jmJobIndex** values.

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

938 In some configurations there may be more than one application program that monitors the
 939 same job when the job passes from one network entity to another when it is submitted.
 940 See configuration 3. When there are multiple job submission IDs, each entity MAY
 941 supply an appropriate **jmJobSubmissionID** value. In this case there would be a separate
 942 entry in the **jmJobSubmissionID** table, one for each **jmJobSubmissionID**. All entries

943 would map to the same **jmJobIndex** that contains the job data. When the job is deleted,
944 it is up to the agent to remove all entries that point to the job from the
945 **jmJobSubmissionID** table as well.

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

949 3.6 Internationalization Considerations

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

951 3.6.1 Text generated by the server or device

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

- 956 1. **jmGeneralJobSetName** object
- 957 2. **processingMessage(6)** attribute
- 958 3. **physicalDevice(32)** (name value) attribute

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

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

965 The text contained in the **processingMessage(6)** attribute is generated by the
966 server/device. The natural language for the **processingMessage(6)** attribute is identified
967 by the **processingMessageNaturalLanguageTag(7)** attribute. The
968 **processingMessageNaturalLanguageTag(7)** attribute uses the
969 **JmNaturalLanguageTagTC** textual convention which SHALL conform to the language
970 tag mechanism specified in RFC 1766 [RFC-1766]. The **JmNaturalLanguageTagTC**
971 value is the same as the IPP [IPP-model] **naturalLanguage**' attribute syntax. RFC 1766
972 specifies that a US-ASCII string consisting of the natural language followed by an
973 optional country field. Both fields use the same two-character codes from ISO 639 [ISO-
974 639] and ISO 3166 [ISO-3166], respectively, that are used in the Printer MIB for
975 identifying language and country.

976 Examples of the values of the **processingMessageNaturalLanguageTag(7)** attribute
977 include:

- 978 1. 'en' for English

- 979 2. 'en-us' for US English
980 3. 'fr' for French
981 4. 'de' for German

982 3.6.2 Text supplied by the job submitter

983 All of the objects and attributes represented by the **JmJobStringTC**' textual-convention
984 are either (1) supplied in the job submission protocol by the client that submits the job to
985 the server or device or (2) are defaulted by the server or device if the job submitting client
986 does not supply values. The agent SHALL represent these objects and attributes in the
987 MIB either (1) in the coded character set as they were submitted or (2) MAY convert the
988 coded character set to another coded character set or encoding scheme. In any case, the
989 resulting coded character set representation SHOULD be UTF-8 [UTF-8], but SHALL be
990 one in which the code positions from 0 to 31 SHALL not be used, 32 to 127 SHALL be
991 US-ASCII [US-ASCII], 127 SHALL be unused, and the remaining code positions 128 to
992 255 SHALL represent single-byte or multi-byte graphic characters structured according to
993 ISO 2022 [ISO 2022] or SHALL be unused.

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

999 Examples of coded character sets which meet this criteria for use as the value of the
1000 **jobCodedCharSet** job attribute are: US-ASCII [US-ASCII], ISO 8859-1 (Latin-1) [ISO
1001 8859-1], any ISO 8859-n, HP Roman8, IBM Code Page 850, Windows Default 8-bit set,
1002 UTF-8 [UTF-8], US-ASCII plus JIS X0208-1990 Japanese [JIS X0208], US-ASCII plus
1003 GB2312-1980 PRC Chinese [GB2312]. See the IANA registry of coded character sets
1004 [IANA charsets].

1005 Examples of coded character sets which do not meet this criteria are: national 7-bit sets
1006 conforming to ISO 646 (except US-ASCII), EBCDIC, and ISO 10646 (Unicode) [ISO-
1007 10646]. In order to represent Unicode characters, the UTF-8 [UTF-8] encoding scheme
1008 SHALL be used which has been assigned the MIBenum value of '106' by IANA.

1009 The **jobCodedCharSet** attribute uses the imported '**CodedCharSet**' textual-convention
1010 from the Printer MIB [printmib].

1011 The natural language for attributes represented by the textual-convention
1012 **JmJobStringTC** SHALL be identified either (1) by the **jobNaturalLanguageTag(9)**
1013 attribute or SHALL be keywords in US-English (as in IPP). A monitoring application
1014 SHOULD attempt to localize keywords into the language of the user by means of some
1015 lookup mechanism. If the keyword value is not known to the monitoring application, the
1016 monitoring application SHOULD assume that the value is in the natural language

1017 specified by the job's **jobNaturalLanguageTag(9)** attribute and SHOULD present the
1018 value to its user as is. The **jobNaturalLanguageTag(9)** attribute value SHALL have the
1019 same syntax and semantics as the **processingMessageNaturalLanguageTag(7)** attribute,
1020 except that the **jobNaturalLanguageTag(9)** attribute identifies the natural language of
1021 attributes supplied by the job submitter instead of the natural language of the
1022 **processingMessage(6)** attribute. See Section 0.

1023 **3.6.3 'DateAndTime' for representing the date and time**

1024 This MIB also contains objects that are represented using the **DateAndTime** textual
1025 convention from SMIV2 [SMIV2-TC]. The job management application SHALL display
1026 such objects in the locale of the user running the monitoring application.

1027 **3.7 IANA and PWG Registration Considerations**

1028 This MIB does not require any additional registration schemes for IANA, but does
1029 depend on registration schemes that other Internet standards track specifications have set
1030 up. The names of these IANA registration assignments under the /in-
1031 notes/iana/assignments/ path:

- 1032 1. printer-language-numbers - used as enums in the **documentFormat(38)** attribute
- 1033 2. media-types - uses as keywords in the **documentFormat(38)** attribute
- 1034 3. character-sets - used as enums in the **jobCodedCharSet(8)** attribute

1035 During the development of this standard, the Printer Working Group (PWG) will register
1036 additional enums while the standard is in the proposed and draft states according to the
1037 procedures described in this section. The PWG will handle registration of additional
1038 enums after approving this standard is approved according to the procedures described in
1039 this section:

1040 **3.7.1 PWG Registration of enums**

1041 This specification uses textual conventions to define enumerated values (enums) and bit
1042 values. Enumerations (enums) and bit values are sets of symbolic values defined for use
1043 with one or more objects or attributes. All enumeration sets and bit value sets are
1044 assigned a symbolic data type name (textual convention). As a convention the symbolic
1045 name ends in "TC" for textual convention. These enumerations are defined at the
1046 beginning of the MIB module specification.

1047 The PWG has defined several type of enumerations for use in the Job Monitoring MIB
1048 and the Printer MIB[print-mib]. These types differ in the method employed to control the
1049 addition of new enumerations. Throughout this document, references to "type n enum",

1050 where n can be 1, 2 or 3 can be found in the various tables. The definitions of these types
1051 of enumerations are:

1052 3.7.1.1 Type 1 enumerations

1053 Type 1 enumeration: All the values are defined in the Job Monitoring MIB specification
1054 (RFC for the Job Monitoring MIB). Additional enumerated values require a new RFC.

1055 There are no type 1 enums in the current draft.

1056 3.7.1.2 Type 2 enumerations

1057 Type 2 enumeration: An initial set of values are defined in the Job Monitoring MIB
1058 specification. Additional enumerated values are registered with the PWG.

1059 The following type 2 enums are contained in the current draft :

- 1060 1. **JmUTF8StringTC**
- 1061 2. **JmJobStringTC**
- 1062 3. **JmNaturalLanguageTagTC**
- 1063 4. **JmTimeStampTC**
- 1064 5. **JmFinishingTC** [same enum values as IPP "finishing" attribute]
- 1065 6. **JmPrintQualityTC** [same enum values as IPP "print-quality" attribute]
- 1066 7. **JmTonerEconomyTC**
- 1067 8. **JmMediumTypeTC**
- 1068 9. **JmJobSubmissionIDTypeTC**
- 1069 10. **JmJobCollationTypeTC**
- 1070 11. **JmJobStateTC** [same enum values as IPP "job-state" attribute]
- 1071 12. **JmAttributeTypeTC**

1072 For those textual conventions that have the same enum values as the indicated IPP Job
1073 attribute SHALL be simultaneously registered by the PWG for use with IPP [ipp-model]
1074 and the Job Monitoring MIB.

1075 3.7.1.3 Type 3 enumeration

1076 Type 3 enumeration: An initial set of values are defined in the Job Monitoring MIB
1077 specification. Additional enumerated values are registered through the PWG without
1078 PWG review.

1079 There are no type 3 enums in the current draft.

1080 3.7.2 PWG Registration of type 2 bit values

1081 This draft contains the following type 2 bit value textual-conventions:

- 1082 1. **JmJobServiceTypesTC**
- 1083 2. **JmJobStateReasons1TC**
- 1084 3. **JmJobStateReasons2TC**
- 1085 4. **JmJobStateReasons3TC**

1086 5. JmJobStateReasons4TC

1087 These textual-conventions are defined as bits in an Integer so that they can be used with
1088 SNMPv1 SMI. The **jobStateReasonsN** ($N=1..4$) attributes are defined as bit values using
1089 the corresponding **JmJobStateReasonsN**TC textual-conventions.

1090 The registration of **JmJobServiceTypesTC** and **JmJobStateReasonsN**TC bit values
1091 SHALL follow the procedures for a type 2 enum as specified in Section 0.

1092 **3.7.3 PWG Registration of Job Submission Id Formats**

1093 In addition to enums and bit values, this specification assigns a single ASCII digit or
1094 letter to various job submission ID formats. See the **JmJobSubmissionIDTypeTC**
1095 textual-convention and the object. The registration of **jmJobSubmissionID** format
1096 numbers SHALL follow the procedures for a type 2 enum as specified in Section 0.

1097 **3.7.4 PWG Registration of MIME types/sub-types for document-formats**

1098 The **documentFormat(38)** attribute has MIME type/sub-type values for indicating
1099 document formats which IANA registers as "media type" names. The values of the
1100 **documentFormat(38)** attribute are the same as the corresponding Internet Printing
1101 Protocol (IPP) "document-format" Job attribute values [ipp-model].

1102 **3.8 Security Considerations**1103 **3.8.1 Read-Write objects**

1104 All objects are read-only, greatly simplifying the security considerations. If another MIB
1105 augments this MIB, that MIB might accept SNMP Write operations to objects in that
1106 MIB whose effect is to modify the values of read-only objects in this MIB. However, that
1107 MIB SHALL have to support the required access control in order to achieve security, not
1108 this MIB.

1109 **3.8.2 Read-Only Objects In Other User's Jobs**

1110 The security policy of some sites MAY be that unprivileged users can only get the objects
1111 from jobs that they submitted, plus a few minimal objects from other jobs, such as the
1112 **jmJobKOctetsPerCopyRequested** and **jmJobKOctetsProcessed** objects, so that a user
1113 can tell how busy a printer is. Other sites MAY allow all unprivileged users to see all
1114 objects of all jobs. This MIB does not require, nor does it specify how, such restrictions
1115 would be implemented. A monitoring application SHOULD enforce the site security
1116 policy with respect to returning information to an unprivileged end user that is using the

1117 monitoring application to monitor jobs that do not belong to that user, i.e., the
1118 **jmJobOwner** object in the **jmJobTable** does not match the user's user name.

1119 An operator is a privileged user that would be able to see all objects of all jobs,
1120 independent of the policy for unprivileged users.

1121 **3.9 Notifications**

1122 This MIB does not specify any notifications. For simplicity, management applications are
1123 expected to poll for status. The **jmGeneralJobPersistence** and
1124 **jmGeneralAttributePersistence** objects assist an application to determine the polling
1125 rate. The resulting network traffic is not expected to be significant.

1126 **4. MIB specification**

1127 The following pages constitute the actual Job Monitoring MIB.

```

1128 Job-Monitoring-MIB DEFINITIONS ::= BEGIN
1129
1130 IMPORTS
    MODULE-IDENTITY, OBJECT-TYPE, enterprises, Integer32
    TEXTUAL-CONVENTION
    MODULE-COMPLIANCE, OBJECT-GROUP
    -- 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
1131
1132 -- Use the enterprises arc assigned to the PWG which is pwg(2699)
1133 -- and assign the first value: jobmon(1) immediately under pwg(2699).
1134
1135 jobmonMIB MODULE-IDENTITY
1136     LAST-UPDATED "9712110000Z"
1137     ORGANIZATION "Printer Working Group (PWG)"
1138     CONTACT-INFO
1139         "Tom Hastings
1140         Postal: Xerox Corp.
1141             Mail stop ESAE-231
1142             701 S. Aviation Blvd.
1143             El Segundo, CA 90245
1144
1145         Tel: (301)333-6413
1146         Fax: (301)333-5514
1147         E-mail: hastings@cp10.es.xerox.com
1148
1149         Send comments to the Printer Working Group (PWG)
1150         using the Job Monitoring Project (JMP) Mailing List:
1151
1152             jmp@pwg.org
1153
1154         For further information, including how to subscribe to the
1155         jmp mailing list, access the PWG web page under 'JMP':
1156         http://www.pwg.org/"
1157     DESCRIPTION
1158         "The MIB module for monitoring job in servers, printers, and other devices.
1159
1160         Version: 1.0"
1161     ::= { enterprises pwg(2699) jobmon(1) }
1162
1163
1164

```

1165 -- Textual conventions for this MIB module

1166

1167

1168

1169 **JmUTF8StringTC** ::= TEXTUAL-CONVENTION

1170 DISPLAY-HINT "255a"

1171 STATUS current

1172 DESCRIPTION

1173 "To facilitate internationalization, this TC represents information taken from the ISO/IEC IS
1174 10646-1 character set, encoded as an octet string using the UTF-8 character encoding scheme."

1175 REFERENCE

1176 "See section 0, entitled: 'Text generated by the server or device'."

1177 SYNTAX OCTET STRING (SIZE (0..63))

1178

1179

1180

1181

1182 **JmJobStringTC** ::= TEXTUAL-CONVENTION

1183 STATUS current

1184 DESCRIPTION

1185 "To facilitate internationalization, this TC represents information using any coded character set
1186 registered by IANA as specified in section IANA and PWG Registration Considerations. While
1187 it is recommended that the coded character set be UTF-8 [UTF-8], the actual coded character
1188 set SHALL be indicated by the value of the **jobCodedCharSet(8)** attribute for the job."

1189 REFERENCE

1190 "See section 0, entitled: 'Text supplied by the job submitter'."

1191 SYNTAX OCTET STRING (SIZE (0..63))

1192

1193

1194

1195

1196 **JmNaturalLanguageTagTC** ::= TEXTUAL-CONVENTION

1197 STATUS current

1198 DESCRIPTION

1199 "An IETF RFC 1766-compliant 'language tag', with zero or more sub-tags that identify a natural
1200 language. While RFC 1766 specifies that the US-ASCII values are case-insensitive, this MIB
1201 specification requires that all characters SHALL be lower case in order to simplify comparing
1202 by management applications."

1203 REFERENCE

1204 "See section 0, entitled: 'Text generated by the server or device' and section 0, entitled: 'Text
1205 supplied by the job submitter'."

1206 SYNTAX OCTET STRING (SIZE (0..63))

1207

1208

1209

1210 **JmTimeStampTC** ::= TEXTUAL-CONVENTION

1211 STATUS current
1212 DESCRIPTION
1213 "The simple time at which an event took place. The units SHALL be in seconds since the
1214 system was booted.
1215
1216 NOTE - **JmTimeStampTC** is defined in units of seconds, rather than 100ths of seconds, so as
1217 to be simpler for agents to implement (even if they have to implement the 100ths of a second to
1218 comply with implementing **sysUpTime** in MIB-II[mib-II].)
1219
1220 NOTE - **JmTimeStampTC** is defined as an **Integer32** so that it can be used as a value of an
1221 attribute, i.e., as a value of the **jmAttributeValueAsInteger** object. The **TimeStamp** textual-
1222 convention defined in SNMPv2-TC [SMIV2-TC] is defined as an **APPLICATION 3**
1223 **IMPLICIT INTEGER** tag, not an **Integer32** which is defined in SNMPv2-SMI [SMIV2-TC]
1224 as UNIVERSAL 2 IMPLICIT INTEGER, so cannot be used in this MIB as one of the values of
1225 **jmAttributeValueAsInteger**."
1226 SYNTAX INTEGER(0..2147483647)
1227
1228
1229
1230
1231 **JmJobSourcePlatformTypeTC ::= TEXTUAL-CONVENTION**
1232 STATUS current
1233 DESCRIPTION
1234 "The source platform type that can submit jobs to servers or devices in any of the 3
1235 configurations."
1236 REFERENCE
1237 "This is a type 2 enumeration. See Section 0. See also IANA operating-system-names
1238 registry."
1239 SYNTAX INTEGER {
1240 **other(1),**
1241 **unknown(2),**
1242 **sptUNIX(3),** -- UNIX
1243 **sptOS2(4),** -- OS/2
1244 **sptPCDOS(5),** -- DOS
1245 **sptNT(6),** -- NT
1246 **sptMVS(7),** -- MVS
1247 **sptVM(8),** -- VM
1248 **sptOS400(9),** -- OS/400
1249 **sptVMS(10),** -- VMS
1250 **sptWindows(11),** -- Windows
1251 **sptNetWare(12)** -- NetWare
1252 }
1253
1254
1255

1246 **JmFinishingTC** ::= TEXTUAL-CONVENTION
1247 STATUS current
1248 DESCRIPTION
1249 "The type of finishing operation.
1250
1251 These values are the same as the enum values of the IPP 'finishings' attribute. See Section 0.
1252
1253 **other(1)**,
1254 Some other finishing operation besides one of the specified or registered values.
1255
1256 **unknown(2)**,
1257 The finishing is unknown.
1258
1259 **none(3)**,
1260 Perform no finishing.
1261
1262 **staple(4)**,
1263 Bind the document(s) with one or more staples. The exact number and placement of the
1264 staples is site-defined.
1265
1266 **punch(5)**,
1267 This value indicates that holes are required in the finished document. The exact number
1268 and placement of the holes is site-defined. The punch specification MAY be satisfied (in
1269 a site- and implementation-specific manner) either by drilling/punching, or by
1270 substituting pre-drilled media.
1271
1272 **cover(6)**,
1273 This value is specified when it is desired to select a non-printed (or pre-printed) cover for
1274 the document. This does not supplant the specification of a printed cover (on cover stock
1275 medium) by the document itself.
1276
1277 **bind(7)**
1278 This value indicates that a binding is to be applied to the document; the type and
1279 placement of the binding is product-specific."
1280 REFERENCE
1281 "This is a type 2 enumeration. See Section 0."
1282 SYNTAX INTEGER {
1283 other(1),
1284 unknown(2),
1285 none(3),
1286 staple(4),
1287 punch(5),
1288 cover(6),
1289 bind(7)
1290 }
1291
1292
1293
1294

1295

1296 **JmPrintQualityTC** ::= TEXTUAL-CONVENTION

1297 STATUS current

1298 DESCRIPTION

1299 "Print quality settings.

1300

1301 These values are the same as the enum values of the IPP 'print-quality' attribute. See Section

1302 0."

1303 REFERENCE

1304 "This is a type 2 enumeration. See Section 0."

1305 SYNTAX INTEGER {

other(1), -- Not one of the specified or registered values.

 --

unknown(2), -- The actual value is unknown.

draft(3), -- Lowest quality available on the printer.

normal(4), -- Normal or intermediate quality on the printer.

 --

high(5) -- Highest quality available on the printer.

1306 }

1307

1308

1309

1310

1311 **JmPrinterResolutionTC** ::= TEXTUAL-CONVENTION

1312 STATUS current

1313 DESCRIPTION

1314 "Printer resolutions.

1315

1316 Nine octets consisting of two 4-octet SIGNED-INTEGERS followed by a SIGNED-BYTE. The

1317 values are the same as those specified in the Printer MIB [printmib]. The first SIGNED-

1318 INTEGER contains the value of prtMarkerAddressabilityXFeedDir. The second SIGNED-

1319 INTEGER contains the value of prtMarkerAddressabilityFeedDir. The SIGNED-BYTE

1320 contains the value of prtMarkerAddressabilityUnit.

1321

1322 Note: the latter value is either 3 (tenThousandsOfInches) or 4 (micrometers) and the

1323 addressability is in 10,000 units of measure. Thus the SIGNED-INTEGERS represent integral

1324 values in either dots-per-inch or dots-per-centimeter.

1325

1326 The syntax is the same as the IPP 'printer-resolution' attribute. See Section 0."

1327 SYNTAX OCTET STRING (SIZE(9))

1328

1329

1330

1331

1332

1333 **JmTonerEconomyTC** ::= TEXTUAL-CONVENTION

1334 STATUS current

1335 DESCRIPTION
1336 "Toner economy settings."
1337 REFERENCE
1338 "This is a type 2 enumeration. See Section 0."
1339 SYNTAX INTEGER {
 unknown(2), -- unknown.
 off(3), -- Off. Normal. Use full toner.
 on(4), -- On. Use less toner than normal.
1340 }
1341
1342
1343
1344
1345
1346 **JmBooleanTC** ::= TEXTUAL-CONVENTION
1347 STATUS current
1348 DESCRIPTION
1349 "Boolean true or false value."
1350 REFERENCE
1351 "This is a type 2 enumeration. See Section 0."
1352 SYNTAX INTEGER {
 unknown(2), -- unknown.
 false(3), -- FALSE.
 true(4), -- TRUE.
1353 }
1354
1355
1356
1357
1358
1359 **JmMediumTypeTC** ::= TEXTUAL-CONVENTION
1360 STATUS current
1361 DESCRIPTION
1362 "Identifies the type of medium.
1363
1364 **other(1)**,
1365 The type is neither one of the values listed in this specification nor a registered value.
1366
1367 **unknown(2)**,
1368 The type is not known.
1369
1370 **stationery(3)**,
1371 Separately cut sheets of an opaque material.
1372
1373 **transparency(4)**,
1374 Separately cut sheets of a transparent material.
1375

1376 **envelope(5),**
 1377 Envelopes that can be used for conventional mailing purposes.
 1378

1379 **envelopePlain(6),**
 1380 Envelopes that are not preprinted and have no windows.
 1381

1382 **envelopeWindow(7),**
 1383 Envelopes that have windows for addressing purposes.
 1384

1385 **continuousLong(8),**
 1386 Continuously connected sheets of an opaque material connected along the long edge.
 1387

1388 **continuousShort(9),**
 1389 Continuously connected sheets of an opaque material connected along the short edge.
 1390

1391 **tabStock(10),**
 1392 Media with tabs.
 1393

1394 **multiPartForm(11),**
 1395 Form medium composed of multiple layers not pre-attached to one another; each sheet
 1396 MAY be drawn separately from an input source.
 1397

1398 **labels(12),**
 1399 Label-stock.
 1400

1401 **multiLayer(13)**
 1402 Form medium composed of multiple layers which are pre-attached to one another, e.g. for
 1403 use with impact printers."
 1404

1405 REFERENCE
 1406 "This is a type 2 enumeration. See Section 0. These enum values correspond to the keyword
 1407 name strings of the **prtInputMediaType** object in the Printer MIB [print-mib]. There is no
 1408 printer description attribute in IPP/1.0 that represents these values."
 1409 SYNTAX INTEGER {

1410 other(1),
 1411 unknown(2),
 1412 stationery(3),
 1413 transparency(4),
 1414 envelope(5),
 1415 envelopePlain(6),
 1416 envelopeWindow(7),
 1417 continuousLong(8),
 1418 continuousShort(9),
 1419 tabStock(10),
 1420 multiPartForm(11),
 1421 labels(12),
 1422 multiLayer(13)
 1423 }

1424

1425

1426

1427

1428 **JmJobCollationTypeTC ::= TEXTUAL-CONVENTION**

1429 STATUS current

1430 DESCRIPTION

1431 "This value is the type of job collation. Implementations that don't support multiple documents
 1432 or don't support multiple copies SHALL NOT support the uncollatedDocuments(5) value.

1433 REFERENCE

1434 "This is a type 2 enumeration. See Section 0. See also Section 0, entitled 'Monitoring Job
 1435 Progress'."

1436 SYNTAX INTEGER {

1437 other(1),

1438 unknown(2),

1439 uncollatedSheets(3),

-- sheets within each document copy

1440 collatedDocuments(4),

-- are not collated: 1 1 ..., 2 2 ...,

1441 uncollatedDocuments(5)

-- internal collated sheets,

1442 uncollatedDocuments(5)

-- documents: A, B, A, B, ...

1443 uncollatedDocuments(5)

-- internal collated sheets,

1444 uncollatedDocuments(5)

-- documents: A, A, ..., B, B, ...

1445 }

1446

1447

1448

1449 **JmJobSubmissionIDTypeTC ::= TEXTUAL-CONVENTION**

1450 STATUS current

1451 DESCRIPTION

1452 "Identifies the format type of a job submission ID.

1453

1454 Each job submission ID is a fixed-length, 48-octet printable US-ASCII [US-ASCII] coded
 1455 character string containing no control characters, consisting of the following fields:

1456

1457 octet 1: The format letter identifying the format. The US-ASCII characters '0-9', 'A-Z', and
 1458 'a-z' are assigned in order giving 62 possible formats.

1459 octets 2-40: A 39-character, US-ASCII trailing SPACE filled field specified by the format
 1460 letter, if the data is less than 39 ASCII characters.

1461 octets 41-48: A sequential or random US-ASCII number to make the ID quasi-unique.

1462

1463 If the client does not supply a job submission ID in the job submission protocol, then the agent
 1464 SHALL assign a job submission ID using any of the standard formats that are reserved for the
 1465 agent. Clients SHALL not use formats that are reserved for agents and agents SHALL NOT use
 1466 formats that are reserved for clients, in order to reduce conflicts in ID generation. See the
 1467 description for which formats are reserved for clients or for agents.

1468

1469 Registration of additional formats may be done following the procedures described in Section 0.

1470

1471 The format values defined at the time of completion of this specification are:

1472

1473	Format
1474	Letter Description
1475	-----
1476	'0' Job Owner generated by the server/device
1477	octets 2-40: The last 39 bytes of the jmJobOwner object.
1478	octets 41-48: The US-ASCII 8-decimal-digit sequential number assigned by the agent.
1479	This format is reserved for agents.
1480	
1481	NOTE - Clients wishing to use a job submission ID that incorporates the job owner, SHALL
1482	use format '8', not format '0'.
1483	
1484	'1' Job Name
1485	octets 2-40: The last 39 bytes of the jobName attribute.
1486	octets 41-48: The US-ASCII 8-decimal-digit random number assigned by the client.
1487	This format is reserved for clients.
1488	
1489	'2' Client MAC address
1490	octets 2-40: The client MAC address: in hexadecimal with each nibble of the 6 octet address
1491	being '0'-'9' or 'A' - 'F' (uppercase only). Most significant octet first.
1492	octets 41-48: The US-ASCII 8-decimal-digit sequential number assigned by the client.
1493	This format is reserved for clients.
1494	
1495	'3' Client URL
1496	octets 2-40: The last 39 bytes of the client URL [URI-spec].
1497	octets 41-48: The US-ASCII 8-decimal-digit sequential number assigned by the client.
1498	This format is reserved for clients.
1499	
1500	'4' Job URI
1501	octets 2-40: The last 39 bytes of the URI [URI-spec] assigned by the server or device to the job
1502	when the job was submitted for processing.
1503	octets 41-48: The US-ASCII 8-decimal-digit sequential number assigned by the agent.
1504	This format is reserved for agents.
1505	
1506	'5' POSIX User Number
1507	octets 2-40: The last 39 bytes of a user number, such as POSIX user number.
1508	octets 41-48: The US-ASCII 8-decimal-digit sequential number assigned by the client.
1509	This format is reserved for clients.
1510	
1511	'6' User Account Number
1512	octets 2-40: The last 39 bytes of the user account number.
1513	octets 41-48: The US-ASCII 8-decimal-digit sequential number assigned by the client.
1514	This format is reserved for clients.
1515	
1516	'7' DTMF Incoming FAX routing number
1517	octets 2-40: The last 39 bytes of the DTMF incoming FAX routing number.
1518	octets 41-48: The US-ASCII 8-decimal-digit sequential number assigned by the client.
1519	This format is reserved for clients.
1520	
1521	'8' Job Owner supplied by the client

- 1522 octets 2-40: The last 39 bytes of the job owner name (that the agent returns in the
1523 **jmJobOwner** object).
- 1524 octets 41-48: The US-ASCII 8-decimal-digit sequential number assigned by the client.
1525 This format is reserved for clients. See format '0' which is reserved for agents.
1526
- 1527 **'9'** Host Name
- 1528 octets 2-40: The last 39 bytes of the host name with trailing SPACES that submitted the job to
1529 this server/device using a protocol, such as LPD [RFC-1179] which includes the host
1530 name in the job submission protocol.
- 1531 octets 41-48: The US-ASCII 8-decimal-digit leading zero representation of the job id generated
1532 by the submitting server (configuration 3) or the client (configuration 1 and 2), such as in
1533 the LPD protocol.
1534 This format is reserved for clients.
1535
- 1536 **'A'** AppleTalk Protocol
- 1537 octets 2-40: Contains the AppleTalk printer name, with the first character of the name in octet
1538 2. AppleTalk printer names are a maximum of 31 characters. Any unused portion of this
1539 field shall be filled with spaces.
- 1540 octets 41-48: '00000XXX', where 'XXX' is the 3-digit US-ASCII decimal representation of the
1541 Connection Id.
1542 This format is reserved for agents.
1543
- 1544 **'B'** NetWare PServer
- 1545 octets 2-40: Contains the Directory Path Name as recorded by the Novell File Server in the
1546 queue directory. If the string is less than 40 octets, the left-most character in the string
1547 shall appear in octet position 2. Otherwise, only the last 39 bytes shall be included. Any
1548 unused portion of this field shall be filled with spaces.
- 1549 octets 41-48: '000XXXXX' The US-ASCII representation of the Job Number as per the
1550 NetWare File Server Queue Management Services.
1551 This format is reserved for agents.
1552
- 1553 **'C'** Server Message Block protocol (SMB)
- 1554 octets 2-40: Contains a decimal (US-ASCII coded) representation of the 16 bit SMB Tree Id
1555 field, which uniquely identifies the connection that submitted the job to the printer. The
1556 most significant digit of the numeric string shall be placed in octet position 2. All unused
1557 portions of this field shall be filled with spaces. The SMB Tree Id has a maximum value
1558 of 65,535.
- 1559 octets 41-48: The US-ASCII 8-decimal-digit leading zero representation of the File Handle
1560 returned from the device to the client in response to a Create Print File command.
1561 This format is reserved for agents.
1562
- 1563 **'D'** Transport Independent Printer/System Interface (TIP/SI)
- 1564 octets 2-40: Contains the Job Name from the Job Control-Start Job (JC-SJ) command. If the
1565 Job Name portion is less than 40 octets, the left-most character in the string shall appear
1566 in octet position 2. Any unused portion of this field shall be filled with spaces.
1567 Otherwise, only the last 39 bytes shall be included.
- 1568 octets 41-48: The US-ASCII 8-decimal-digit leading zero representation of the **jmJobIndex**
1569 assigned by the agent.

1570 This format is reserved for agents, since the agent supplies octets 41-48, though the client
 1571 supplies the job name. See format '1' reserved to clients to submit job name ids in which
 1572 they supply octets 41-48.
 1573

1574 NOTE - the job submission id is only intended to be unique between a limited set of clients for
 1575 a limited duration of time, namely, for the life time of the job in the context of the server or
 1576 device that is processing the job. Some of the formats include something that is unique per
 1577 client and a random number so that the same job submitted by the same client will have a
 1578 different job submission id. For other formats, where part of the id is guaranteed to be unique
 1579 for each client, such as the MAC address or URL, a sequential number SHOULD suffice for
 1580 each client (and may be easier for each client to manage). Therefore, the length of the job
 1581 submission id has been selected to reduce the probability of collision to an extremely low
 1582 number, but is not intended to be an absolute guarantee of uniqueness. None-the-less,
 1583 collisions are remotely possible, but without bad consequences, since this MIB is intended to be
 1584 used only for monitoring jobs, not for controlling and managing them."

1585 REFERENCE

1586 "This is like a type 2 enumeration. See section 0."

1587 SYNTAX OCTET STRING(SIZE(1)) -- ASCII '0'-'9', 'A'-'Z', 'a'-'z'
 1588
 1589
 1590
 1591
 1592

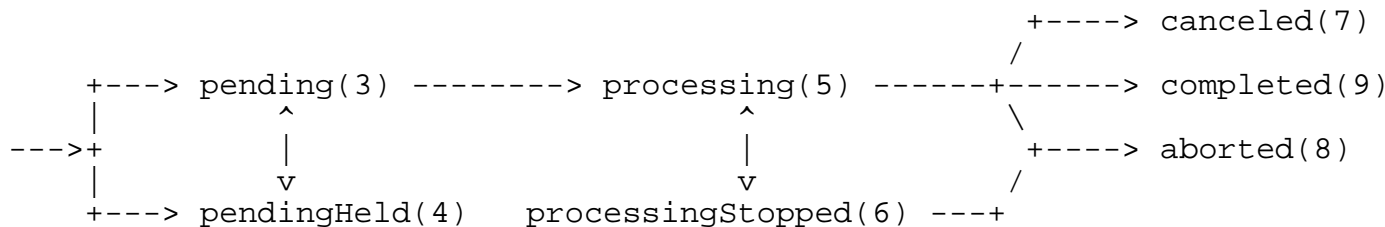
1593 **JmJobStateTC** ::= TEXTUAL-CONVENTION

1594 STATUS current

1595 DESCRIPTION

1596 "The current state of the job (**pending**, **processing**, **completed**, etc.).
 1597

1598 The following figure shows the normal job state transitions:
 1599



1608 **Figure 4 - Normal Job State Transitions**

1609 Normally a job progresses from left to right. Other state transitions are unlikely, but are not
 1610 forbidden. Not shown are the transitions to the **canceled** state from the **pending**,
 1611 **pendingHeld**, and **processingStopped** states.
 1612

1613 Jobs in the **pending**, **processing**, and **processingStopped** states are called 'active', while jobs in
 1614 the **pendingHeld**, **canceled**, **aborted**, and **completed** states are called 'inactive'. Jobs reach
 1615 one of the three terminal states: **completed**, **canceled**, or **aborted**, *after* the jobs have
 1616

1617 completed all activity, and all MIB objects and attributes have reached their final values for the
1618 job.

1619
1620 These values are the same as the enum values of the IPP 'job-state' job attribute. See Section0.

1621
1622 **unknown(2),**

1623 The job state is *not* known, or its state is indeterminate.

1624
1625 **pending(3),**

1626 The job is a candidate to start processing, but is not yet processing.

1627
1628 **pendingHeld(4),**

1629 The job is not a candidate for processing for any number of reasons but will return to the
1630 **pending** state as soon as the reasons are no longer present. The job's
1631 **jmJobStateReasons1** object and/or **jobStateReasonsN** ($N=2..4$) attributes SHALL
1632 indicate why the job is no longer a candidate for processing. The reasons are represented
1633 as bits in the **jmJobStateReasons1** object and/or **jobStateReasonsN** ($N=2..4$) attributes.
1634 See the **JmJobStateReasonsN**TC ($N=1..4$) textual convention for the specification of
1635 each reason.

1636
1637 **processing(5),**

1638 One or more of:

1639
1640 1. the job is using, or is attempting to use, one or more purely software processes that are
1641 analyzing, creating, or interpreting a PDL, etc.,

1642
1643 2. the job is using, or is attempting to use, one or more hardware devices that are
1644 interpreting a PDL, making marks on a medium, and/or performing finishing, such as
1645 stapling, etc.,

1646
1647 OR

1648
1649 3. (configuration 2) the server has made the job ready for printing, but the output device is
1650 not yet printing it, either because the job hasn't reached the output device or because the
1651 job is queued in the output device or some other spooler, awaiting the output device to
1652 print it.

1653
1654 When the job is in the **processing** state, the entire job state includes the detailed status
1655 represented in the device MIB indicated by the **hrDeviceIndex** value of the job's
1656 **physicalDevice** attribute, if the agent implements such a device MIB.

1657
1658 Implementations MAY, though they NEED NOT, include additional values in the job's
1659 **jmJobStateReasons1** object to indicate the progress of the job, such as adding the
1660 **jobPrinting** value to indicate when the device is actually making marks on a medium
1661 and/or the **processingToStopPoint** value to indicate that the server or device is in the
1662 process of canceling or aborting the job.

1663

1664 **processingStopped(6),**
 1665 The job has stopped while processing for any number of reasons and will return to the
 1666 **processing** state as soon as the reasons are no longer present.
 1667
 1668 The job's **jmJobStateReasons1** object and/or the job's **jobStateReasonsN** ($N=2..4$)
 1669 attributes MAY indicate why the job has stopped processing. For example, if the output
 1670 device is stopped, the **deviceStopped** value MAY be included in the job's
 1671 **jmJobStateReasons1** object.
 1672
 1673 NOTE - When an output device is stopped, the device usually indicates its condition in
 1674 human readable form at the device. The management application can obtain more
 1675 complete device status remotely by querying the appropriate device MIB using the job's
 1676 **deviceIndex** attribute(s), if the agent implements such a device MIB
 1677
 1678 **canceled(7),**
 1679 A client has canceled the job and the server or device has completed canceling the job
 1680 AND all MIB objects and attributes have reached their final values for the job. While the
 1681 server or device is canceling the job, the job's **jmJobStateReasons1** object SHOULD
 1682 contain the **processingToStopPoint** value and one of the **canceledByUser**,
 1683 **canceledByOperator**, or **canceledAtDevice** values. The **canceledByUser**,
 1684 **canceledByOperator**, or **canceledAtDevice** values remain while the job is in the
 1685 **canceled** state.
 1686
 1687 **aborted(8),**
 1688 The job has been aborted by the system, usually while the job was in the **processing** or
 1689 **processingStopped** state and the server or device has completed aborting the job AND all
 1690 MIB objects and attributes have reached their final values for the job. While the server or
 1691 device is aborting the job, the job's **jmJobStateReasons1** object MAY contain the
 1692 **processingToStopPoint** and **abortedBySystem** values. If implemented, the
 1693 **abortedBySystem** value SHALL remain while the job is in the **aborted** state.
 1694
 1695 **completed(9)**
 1696 The job has completed successfully or with warnings or errors after processing and all of
 1697 the media have been successfully stacked in the appropriate output bin(s) AND all MIB
 1698 objects and attributes have reached their final values for the job. The job's
 1699 **jmJobStateReasons1** object SHOULD contain one of: **completedSuccessfully**,
 1700 **completedWithWarnings**, or **completedWithErrors** values."
 1701 REFERENCE
 1702 "This is a type 2 enumeration. See Section 0."
 1703 SYNTAX INTEGER {
 1704 unknown(2),
 1705 pending(3),
 1706 pendingHeld(4),
 1707 processing(5),
 1708 processingStopped(6),
 1709 canceled(7),
 1710 aborted(8),
 1711 completed(9)
 1712 }

1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760

JmAttributeTypeTC ::= TEXTUAL-CONVENTION

STATUS current

DESCRIPTION

"The type of the attribute which identifies the attribute.

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

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

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

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

Values in the range 2**30 to 2**31-1 are reserved for private or experimental usage. This range corresponds to the same range reserved in IPP. Implementers are warned that use of such values may conflict with other implementations. Implementers are encouraged to request registration of enum values following the procedures in Section 0.

NOTE: No attribute name exceeds 31 characters.

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

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

++++
+ **Job State attributes**
+

1761 + The following attributes specify the state of a job.
 1762 ++++++

1763

1764 **jobStateReasons2(3),** **JmJobStateReasons2TC**
 1765 INTEGER: Additional information about the job's current state that augments the
 1766 **jmJobState** object. See the description under the **JmJobStateReasons1TC** textual-
 1767 convention.
 1768

1769 **jobStateReasons3(4),** **JmJobStateReasons3TC**
 1770 INTEGER: Additional information about the job's current state that augments the
 1771 **jmJobState** object. See the description under **JmJobStateReasons1TC** textual-
 1772 convention.
 1773

1774 **jobStateReasons4(5),** **JmJobStateReasons4TC**
 1775 INTEGER: Additional information about the job's current state that augments the
 1776 **jmJobState** object. See the description under **JmJobStateReasons1TC** textual-
 1777 convention.
 1778

1779 **processingMessage(6),** **JmUTF8StringTC(SIZE(0..63))**
 1780 OCTETS: MULTI-ROW: A coded character set message that is generated by the server
 1781 or device during the processing of the job as a simple form of processing log to show
 1782 progress and any problems. The natural language of each value is specified by the
 1783 corresponding **processingMessageNaturalLanguageTag(7)** value.
 1784

1785 NOTE - This attribute is intended for such conditions as interpreter messages, rather than
 1786 being the printable form of the **jmJobState** and **jmJobStateReasons1** objects and
 1787 **jobStateReasons2**, **jobStateReasons3**, and **jobStateReasons4** attributes. In order to
 1788 produce a localized printable form of these job state objects/attribute, a management
 1789 application SHOULD produce a message from their enum and bit values.
 1790

1791 NOTE - There is no job description attribute in IPP/1.0 that corresponds to this attribute
 1792 and this attribute does not correspond to the IPP/1.0 'job-state-message' job description
 1793 attribute, which is just a printable form of the IPP 'job-state' and 'job-state-reasons' job
 1794 attributes.
 1795

1796 There is no restriction for the same message occurring in multiple rows.
 1797

1798 **processingMessageNaturalLanguageTag(7),** **OCTET STRING(SIZE(0..63))**
 1799 OCTETS: MULTI-ROW: The natural language of the corresponding
 1800 **processingMessage(6)** attribute value. See section 0, entitled 'Text generated by the
 1801 server or device'.
 1802

1803 If the agent does not know the natural language of the job processing message, the agent
 1804 SHALL either (1) return a zero length string value for the
 1805 **processingMessageNaturalLanguageTag(7)** attribute or (2) not return the
 1806 **processingMessageNaturalLanguageTag(7)** attribute for the job.
 1807

1808 There is no restriction for the same tag occurring in multiple rows, since when this
1809 attribute is implemented, it SHOULD have a value row for each corresponding
1810 processingMessage(6) attribute value row.

1811 jobCodedCharSet(8), CodedCharSet

1812 INTEGER: The MIBenum identifier of the coded character set that the agent is using to
1813 represent coded character set objects and attributes of type JmJobStringTC'. These
1814 coded character set objects and attributes are either: (1) supplied by the job submitting
1815 client or (2) defaulted by the server or device when omitted by the job submitting client.
1816 The agent SHALL represent these objects and attributes in the MIB either (1) in the coded
1817 character set as they were submitted or (2) MAY convert the coded character set to
1818 another coded character set or encoding scheme as identified by the
1819 jobCodedCharSet(8) attribute. See section 0, entitled 'Text supplied by the job
1820 submitter'.

1821
1822
1823 These MIBenum values are assigned by IANA [IANA-charsets] when the coded character
1824 sets are registered. The coded character set SHALL be one of the ones registered with
1825 IANA [IANA] and the enum value uses the CodedCharSet textual-convention from the
1826 Printer MIB. See the JmJobStringTC textual-convention.

1827
1828 If the agent does not know what coded character set was used by the job submitting
1829 client, the agent SHALL either (1) return the unknown(2)' value for the
1830 jobCodedCharSet(8) attribute or (2) not return the jobCodedCharSet(8) attribute for
1831 the job.

1832
1833 jobNaturalLanguageTag(9), OCTET STRING(SIZE(0..63))

1834 OCTETS: The natural language of the job attributes supplied by the job submitter or
1835 defaulted by the server or device for the job, i.e., all objects and attributes represented by
1836 the 'JmJobStringTC' textual-convention, such as jobName, mediumRequested, etc.
1837 See Section 0, entitled 'Text supplied by the job submitter'.

1838
1839 If the agent does not know what natural language was used by the job submitting client,
1840 the agent SHALL either (1) return a zero length string value for the
1841 jobNaturalLanguageTag(9) attribute or (2) not return jobNaturalLanguageTag(9)
1842 attribute for the job.

1843
1844
1845 ++++++
1846 + Job Identification attributes
1847 +
1848 + The following attributes help an end user, a system
1849 + operator, or an accounting program identify a job.
1850 ++++++

1851
1852
1853
1854 jobURI(20), OCTET STRING(SIZE(0..63))

1855 OCTETS: MULTI-ROW: The job's Universal Resource Identifier (URI) [RFC-1738].
1856 See IPP [ipp-model] for example usage.

1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905

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 63 octets, the agent SHALL use multiple values, with the next 63 octets coming in the second value, etc.

NOTE - IPP [ipp-model] has a 1023-octet maximum length for a URI, though the URI standard itself and HTTP/1.1 specify no maximum length.

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

NOTE: This attribute NEED NOT be printable characters.

serverAssignedJobName(22), **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

1906		himself/herself, such as a reminder of what to do with the results or to indicate a different
1907		set of input parameters were tried in several different job submissions.
1908		
1909	jobServiceTypes(24),	JmJobServiceTypesTC
1910		INTEGER: Specifies the type(s) of service to which the job has been submitted (print,
1911		fax, scan, etc.). The service type is bit encoded with each job service type so that more
1912		general and arbitrary services can be created, such as services with more than one
1913		destination type, or ones with only a source or only a destination. For example, a job
1914		service might scan , faxOut , and print a single job. In this case, three bits would be set in
1915		the jobServiceTypes attribute, corresponding to the hexadecimal values: 0x8 + 0x20 +
1916		0x4 , respectively, yielding: 0x2C .
1917		
1918		Whether this attribute is set from a job attribute supplied by the job submission client or
1919		is set by the recipient job submission server or device depends on the job submission
1920		protocol. This attribute SHALL be implemented if the server or device has other types in
1921		addition to or instead of printing.
1922		
1923		One of the purposes of this attribute is to permit a requester to filter out jobs that are not
1924		of interest. For example, a printer operator may only be interested in jobs that include
1925		printing.
1926		
1927	jobSourceChannelIndex(25),	Integer32(0..2147483647)
1928		INTEGER: The index of the row in the associated Printer MIB[print-mib] of the channel
1929		which is the source of the print job.
1930		
1931	jobSourcePlatformType(26),	JmJobSourcePlatformTypeTC
1932		INTEGER: The source platform type of the immediate upstream submitter that submitted
1933		the job to the server (configuration 2) or device (configuration 1 and 3) to which the agent
1934		is providing access. For configuration 1, this is the type of the client that submitted the
1935		job to the device; for configuration 2, this is the type of the client that submitted the job
1936		to the server; and for configuration 3, this is the type of the server that submitted the job
1937		to the device.
1938		
1939	submittingServerName(27),	JmJobStringTC(SIZE(0..63))
1940		OCTETS: For configuration 3 only: The administrative name of the server that
1941		submitted the job to the device.
1942		
1943	submittingApplicationName(28),	JmJobStringTC(SIZE(0..63))
1944		OCTETS: The name of the client application (not the server in configuration 3) that
1945		submitted the job to the server or device.
1946		
1947	jobOriginatingHost(29),	JmJobStringTC(SIZE(0..63))
1948		OCTETS: The name of the client host (not the server host name in configuration 3) that
1949		submitted the job to the server or device.
1950		
1951	deviceNameRequested(30),	JmJobStringTC(SIZE(0..63))
1952		OCTETS: The administratively defined coded character set name of the target device
1953		requested by the submitting user. For configuration 1, its value corresponds to the Printer
1954		MIB[print-mib]: prtGeneralPrinterName object. For configuration 2 and 3, its value is

1955 the name of the logical or physical device that the user supplied to indicate to the server
 1956 on which device(s) they wanted the job to be processed.
 1957

1958 **queueNameRequested(31),** **JmJobStringTC(SIZE(0..63))**
 1959 OCTETS: The administratively defined coded character set name of the target queue
 1960 requested by the submitting user. For configuration 1, its value corresponds to the queue
 1961 in the device for which the agent is providing access. For configuration 2 and 3, its value
 1962 is the name of the queue that the user supplied to indicate to the server on which device(s)
 1963 they wanted the job to be processed.
 1964

1965 NOTE - typically an implementation SHOULD support either the **deviceNameRequested**
 1966 or **queueNameRequested** attribute, but not both.
 1967

1968 **physicalDevice(32),** **hrDeviceIndex**
 1969 AND/OR
 1970 **JmUTF8StringTC(SIZE(0..63))**
 1971 INTEGER: MULTI-ROW: The index of the physical device MIB instance
 1972 requested/used, such as the Printer MIB[print-mib]. This value is an **hrDeviceIndex**
 1973 value. See the Host Resources MIB[hr-mib].
 1974

1975 AND/OR
 1976

1977 OCTETS: MULTI-ROW: The name of the physical device to which the job is assigned.
 1978

1979 **numberOfDocuments(33),** **Integer32(-2..2147483647)**
 1980 INTEGER: The number of documents in this job.
 1981

1982 The agent SHOULD return this attribute if the job has more than one document.
 1983

1984 **fileName(34),** **JmJobStringTC(SIZE(0..63))**
 1985 OCTETS: MULTI-ROW: The coded character set file name or URI[URI-spec] of the
 1986 document.
 1987

1988 There is no restriction on the same file name occurring in multiple rows.
 1989

1990 **documentName(35),** **JmJobStringTC(SIZE(0..63))**
 1991 OCTETS: MULTI-ROW: The coded character set name of the document.
 1992

1993 There is no restriction on the same document name occurring in multiple rows.
 1994

1995 **jobComment(36),** **JmJobStringTC(SIZE(0..63))**
 1996 OCTETS: An arbitrary human-readable coded character text string supplied by the
 1997 submitting user or the job submitting application program for any purpose. For example,
 1998 a user might indicate what he/she is going to do with the printed output or the job
 1999 submitting application program might indicate how the document was produced.
 2000

2001 The **jobComment** attribute is not intended to be a name; see the **jobName** attribute.
 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
2047
2048
2049
2050
2051

documentFormatIndex(37),

Integer32(0..2147483647)

INTEGER: MULTI-ROW: The index in the **prtInterpreterTable** in the Printer MIB[print-mib] of the page description language (PDL) or control language interpreter that this job requires/uses. A document or a job MAY use more than one PDL or control language.

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', 'application/pdf', 'text/plain' (US-ASCII SHALL be assumed), 'text/plain; charset=iso-8859-1', and 'application/octet-stream'. The IPP 'document-format' job attribute uses these same values with the same semantics. See the IPP [ipp-model] 'mimeMediaType' attribute syntax and the document-format attribute for further examples and explanation.

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

jobPriority(50),

Integer32(-2..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. -2 indicates unknown.

2052
 2053 **jobProcessAfterDateAndTime(51),** **DateAndTime** (SNMPv2-TC)
 2054 OCTETS: The calendar date and time of day after which the job SHALL become a
 2055 candidate to be scheduled for processing. If the value of this attribute is in the future, the
 2056 server SHALL set the value of the job's **jmJobState** object to **pendingHeld** and add the
 2057 **jobProcessAfterSpecified** bit value to the job's **jmJobStateReasons1** object. When the
 2058 specified date and time arrives, the server SHALL remove the **jobProcessAfterSpecified**
 2059 bit value from the job's **jmJobStateReasons1** object and, if no other reasons remain,
 2060 SHALL change the job's **jmJobState** object to **pending**.
 2061

2062 **jobHold(52),** **JmBooleanTC**
 2063 INTEGER: If the value is **true(4)**, a client has explicitly specified that the job is to be
 2064 held until explicitly released. Until the job is explicitly released by a client, the job
 2065 SHALL be in the **pendingHeld** state with the **jobHoldSpecified** value in the
 2066 **jmJobStateReasons1** attribute.
 2067

2068 **jobHoldUntil(53),** **JmJobStringTC(SIZE(0..63))**
 2069 OCTETS: The named time period during which the job SHALL become a candidate for
 2070 processing, such as **'evening'**, **'night'**, **'weekend'**, **'second-shift'**, **'third-shift'**, etc., as
 2071 defined by the system administrator. See IPP [ipp-model] for the standard keyword
 2072 values. Until that time period arrives, the job SHALL be in the **pendingHeld** state with
 2073 the **jobHoldUntilSpecified** value in the **jmJobStateReasons1** object. The value **'no-**
 2074 **hold'** SHALL indicate explicitly that no time period has been specified; the absence of
 2075 this attribute SHALL indicate implicitly that no time period has been specified.
 2076

2077 **outputBin(54),** **Integer32(0..2147483647)**
 2078 **AND/OR**
 2079 **JmJobStringTC(SIZE(0..63))**
 2080 INTEGER: MULTI-ROW: The output subunit index in the Printer MIB[print-mib]
 2081
 2082 AND/OR
 2083
 2084 OCTETS: MULTI-ROW: the name or number (represented as ASCII digits) of the
 2085 output bin to which all or part of the job is placed in.
 2086

2087 **sides(55),** **Integer32(-2..2)**
 2088 INTEGER: MULTI-ROW: The number of sides, **1** or **2**, that any document in this job
 2089 requires/used.
 2090

2091 **finishing(56),** **JmFinishingTC**
 2092 INTEGER: MULTI-ROW: Type of finishing that any document in this job
 2093 requires/used.
 2094
 2095
 2096 +++++
 2097 + **Image Quality attributes (requested and consumed)**
 2098 +
 2099 + **For devices that can vary the image quality.**
 2100 +++++

2101 **printQualityRequested(70),** **JmPrintQualityTC**
2102 **INTEGER: MULTI-ROW:** The print quality selection requested for a document in the
2103 job for printers that allow quality differentiation.
2104
2105 **printQualityUsed(71),** **JmPrintQualityTC**
2106 **INTEGER: MULTI-ROW:** The print quality selection actually used by a document in
2107 the job for printers that allow quality differentiation.
2108
2109 **printerResolutionRequested(72),** **JmPrinterResolutionTC**
2110 **OCTETS: MULTI-ROW:** The printer resolution requested for a document in the job for
2111 printers that support resolution selection.
2112
2113 **printerResolutionUsed(73),** **JmPrinterResolutionTC**
2114 **OCTETS: MULTI-ROW:** The printer resolution actually used by a document in the job
2115 for printers that support resolution selection.
2116
2117 **tonerEcomonyRequested(74),** **JmTonerEcomonyTC**
2118 **INTEGER: MULTI-ROW:** The toner economy selection requested for documents in the
2119 job for printers that allow toner economy differentiation.
2120
2121 **tonerEcomonyUsed(75),** **JmTonerEcomonyTC**
2122 **INTEGER: MULTI-ROW:** The toner economy selection actually used by documents in
2123 the job for printers that allow toner economy differentiation.
2124
2125 **tonerDensityRequested(76),** **Integer32(-2..100)**
2126 **INTEGER: MULTI-ROW:** The toner density requested for a document in this job for
2127 devices that can vary toner density levels. Level 1 is the lowest density and level 100 is
2128 the highest density level. Devices with a smaller range, SHALL map the 1-100 range
2129 evenly onto the implemented range.
2130
2131 **tonerDensityUsed(77),** **Integer32(-2..100)**
2132 **INTEGER: MULTI-ROW:** The toner density used by documents in this job for devices
2133 that can vary toner density levels. Level 1 is the lowest density and level 100 is the
2134 highest density level. Devices with a smaller range, SHALL map the 1-100 range evenly
2135 onto the implemented range.
2136
2137
2138
2139 ++++++
2140 + **Job Progress attributes (requested and consumed)**
2141 +
2142 + **Pairs of these attributes can be used by monitoring**
2143 + **applications to show an indication of relative progress**
2144 + **to users. See section 0, entitled**
2145 + **'Monitoring Job Progress'.**
2146 ++++++
2147
2148 **jobCopiesRequested(90),** **Integer32(-2..2147483647)**
2149 **INTEGER:** The number of copies of the entire job that are to be produced.

2150
 2151 **jobCopiesCompleted(91),** **Integer32(-2..2147483647)**
 2152 INTEGER: The number of copies of the entire job that have been completed so far.
 2153

2154 **documentCopiesRequested(92),** **Integer32(-2..2147483647)**
 2155 INTEGER: The total count of the number of document copies requested for the job as a
 2156 whole. If there are documents A, B, and C, and document B is specified to produce 4
 2157 copies, the number of document copies requested is 6 for the job.
 2158
 2159 This attribute SHALL be used only when a job has multiple documents. The
 2160 **jobCopiesRequested** attribute SHALL be used when the job has only one document.
 2161
 2162

2163 **documentCopiesCompleted(93),** **Integer32(-2..2147483647)**
 2164 INTEGER: The total count of the number of document copies completed so far for the
 2165 job as a whole. If there are documents A, B, and C, and document B is specified to
 2166 produce 4 copies, the number of document copies starts a 0 and runs up to 6 for the job as
 2167 the job processes.
 2168
 2169 This attribute SHALL be used only when a job has multiple documents. The
 2170 **jobCopiesCompleted** attribute SHALL be used when the job has only one document.
 2171
 2172

2173 **jobKOctetsTransferred(94),** **Integer32(-2..2147483647)**
 2174 INTEGER: The number of K (1024) octets transferred to the server or device to which
 2175 the agent is providing access. This count is independent of the number of copies of the
 2176 job or documents that will be produced, but it is only a measure of the number of bytes
 2177 transferred to the server or device.
 2178
 2179 The agent SHALL round the actual number of octets transferred up to the next higher K.
 2180 Thus 0 octets SHALL be represented as 0', 1-1024 octets SHALL BE represented as 1',
 2181 1025-2048 SHALL be 2', etc. When the job completes, the values of the
 2182 **jmJobKOctetsPerCopyRequested** object and the **jobKOctetsTransferred** attribute
 2183 SHALL be equal.
 2184
 2185 NOTE - The **jobKOctetsTransferred** can be used with the
 2186 **jmJobKOctetsPerCopyRequested** object in order to produce a relative indication of the
 2187 progress of the job for agents that do not implement the **jmJobKOctetsProcessed** object.
 2188

2189 **sheetCompletedCopyNumber(95),** **Integer32(-2..2147483647)**
 2190 INTEGER: The number of the copy being stacked for the current document. This
 2191 number starts at 0, is set to 1 when the first sheet of the first copy for each document is
 2192 being stacked and is equal to n where n is the nth sheet stacked in the current document
 2193 copy. See section Monitoring Job Progress , entitled 'Monitoring Job Progress'.
 2194

2195 **sheetCompletedDocumentNumber(96),** **Integer32(-2..2147483647)**
 2196 INTEGER: The ordinal number of the document in the job that is currently being
 2197 stacked. This number starts at 0, increments to 1 when the first sheet of the first
 2198 document in the job is being stacked, and is equal to n where n is the nth document in the
 job, starting with 1.

2199 Implementations that only support one document jobs SHOULD NOT implement this
2200 attribute.
2201

2202 **jobCollationType(97),** **JmJobCollationTypeTC**
2203 **INTEGER:** The type of job collation. See also Section 0, entitled 'Monitoring Job
2204 Progress'.

2205
2206
2207
2208 ++++++
2209 + **Impression attributes**
2210 +
2211 + **See the definition of the terms 'impression', 'sheet',**
2212 + **and 'page' in Section 0.**
2213 +
2214 + **See also jmJobImpressionsPerCopyRequested and**
2215 + **jmJobImpressionsCompleted objects in the jmJobTable.**
2216 ++++++

2217
2218 **impressionsSpooled(110),** **Integer32(-2..2147483647)**
2219 **INTEGER:** The number of impressions spooled to the server or device for the job so far.

2220
2221 **impressionsSentToDevice(111),** **Integer32(-2..2147483647)**
2222 **INTEGER:** The number of impressions sent to the device for the job so far.

2223
2224 **impressionsInterpreted(112),** **Integer32(-2..2147483647)**
2225 **INTEGER:** The number of impressions interpreted for the job so far.

2226
2227 **impressionsCompletedCurrentCopy(113),** **Integer32(-2..2147483647)**
2228 **INTEGER:** The number of impressions completed by the device for the current copy of
2229 the current document so far. For printing, the impressions completed includes
2230 interpreting, marking, and stacking the output. For other types of job services, the
2231 number of impressions completed includes the number of impressions processed.
2232
2233 This value SHALL be reset to 0 for each document in the job and for each document
2234 copy.

2235
2236 **fullColorImpressionsCompleted(114),** **Integer32(-2..2147483647)**
2237 **INTEGER:** The number of full color impressions completed by the device for this job so
2238 far. For printing, the impressions completed includes interpreting, marking, and stacking
2239 the output. For other types of job services, the number of impressions completed includes
2240 the number of impressions processed. Full color impressions are typically defined as
2241 those requiring 3 or more colorants, but this MAY vary by implementation. In any case,
2242 the value of this attribute counts by 1 for each side that has full color, not by the number
2243 of colors per side (and the other impression counters are incremented, except
2244 **highlightColorImpressionsCompleted(115)).**

2245
2246 **highlightColorImpressionsCompleted(115),** **Integer32(-2..2147483647)**
2247 **INTEGER:** The number of highlight color impressions completed by the device for this

2248 job so far. For printing, the impressions completed includes interpreting, marking, and
 2249 stacking the output. For other types of job services, the number of impressions completed
 2250 includes the number of impressions processed. Highlight color impressions are typically
 2251 defined as those requiring black plus one other colorant, but this MAY vary by
 2252 implementation. In any case, the value of this attribute counts by 1 for each side that has
 2253 highlight color (and the other impression counters are incremented, except
 2254 **fullColorImpressionsCompleted(114)**).

2255
 2256
 2257 ++++++
 2258 + **Page attributes**
 2259 +
 2260 + See the definition of 'impression', 'sheet', and 'page'
 2261 + in Section 0.
 2262 ++++++

2263
 2264 **pagesRequested(130), Integer32(-2..2147483647)**
 2265 INTEGER: The number of logical pages requested by the job to be processed.

2266
 2267 **pagesCompleted(131), Integer32(-2..2147483647)**
 2268 INTEGER: The number of logical pages completed for this job so far.

2269
 2270 For implementations where multiple copies are produced by the interpreter with only a
 2271 single pass over the data, the final value SHALL be equal to the value of the
 2272 **pagesRequested** object. For implementations where multiple copies are produced by the
 2273 interpreter by processing the data for each copy, the final value SHALL be a multiple of
 2274 the value of the **pagesRequested** object.

2275
 2276 NOTE - See the **impressionsCompletedCurrentCopy** and
 2277 **pagesCompletedCurrentCopy** attributes for attributes that are reset on each document
 2278 copy.

2279
 2280 NOTE - The **pagesCompleted** object can be used with the **pagesRequested** object to
 2281 provide an indication of the relative progress of the job, provided that the multiplicative
 2282 factor is taken into account for some implementations of multiple copies.

2283
 2284 **pagesCompletedCurrentCopy(132), Integer32(-2..2147483647)**
 2285 INTEGER: The number of logical pages completed for the current copy of the document
 2286 so far. This value SHALL be reset to 0 for each document in the job and for each
 2287 document copy.

2288
 2289
 2290 ++++++
 2291 + **Sheet attributes**
 2292 +
 2293 + See the definition of 'impression', 'sheet', and 'page'
 2294 + in Section 0.
 2295 ++++++

2296

2297 **sheetsRequested(150),** **Integer32(-2..2147483647)**
2298 INTEGER: The total number of medium sheets requested to be produced for this job.
2299
2300 Unlike the **jmJobKOctetsPerCopyRequested** and
2301 **jmJobImpressionsPerCopyRequested** attributes, the **sheetsRequested(150)** attribute
2302 SHALL include the multiplicative factor contributed by the number of copies and so is
2303 the total number of sheets to be produced by the job, as opposed to the size of the
2304 document(s) submitted.
2305
2306 **sheetsCompleted(151),** **Integer32(-2..2147483647)**
2307 INTEGER: The total number of medium sheets that have completed marking and
2308 stacking for the entire job so far whether those sheets have been processed on one side or
2309 on both.
2310
2311 **sheetsCompletedCurrentCopy(152),** **Integer32(-2..2147483647)**
2312 INTEGER: The number of medium sheets that have completed marking and stacking for
2313 the current copy of a document in the job so far whether those sheets have been processed
2314 on one side or on both.
2315
2316 The value of this attribute SHALL be 0 before the job starts processing and SHALL be
2317 reset to **1** after the first sheet of each document and document copy in the job processed
2318 and stacked.
2319
2320
2321 ++++++
2322 + **Resources attributes (requested and consumed)**
2323 +
2324 + **Pairs of these attributes can be used by monitoring**
2325 + **applications to show an indication of relative usage to**
2326 + **users.**
2327 ++++++

2328
2329 **mediumRequested(170),** **JmMediumTypeTC**
2330 AND/OR
2331 **JmJobStringTC(SIZE(0..63))**
2332 INTEGER: MULTI-ROW: The type
2333 AND/OR
2334 OCTETS: MULTI-ROW: the name of the medium that is required by the job.
2335
2336 NOTE - The name (**JmJobStringTC**) values correspond to the **prtInputMediaName**
2337 object in the Printer MIB [print-mib] and the values of the IPP 'media' attribute.
2338
2339 **mediumConsumed(171),** **Integer32(-2..2147483647)**
2340 AND
2341 **JmJobStringTC(SIZE(0..63))**
2342 INTEGER: The number of sheets
2343 AND
2344 OCTETS: MULTI-ROW: MULTI-ROW: the name of the medium that has been
2345 consumed so far whether those sheets have been processed on one side or on both.

2346 This attribute SHALL have both **Integer32** and **OCTET STRING** (represented as
 2347 **JmJobStringTC**) values.
 2348

2349
 2350 NOTE - The name (**JmJobStringTC**) values correspond to the name values of the
 2351 **prtInputMediaName** object in the Printer MIB [print-mib].
 2352

2353 **colorantRequested(172), Integer32(-2..2147483647)**
 2354 **AND/OR**
 2355 **JmJobStringTC(SIZE(0..63))**
 2356 INTEGER: MULTI-ROW: The index (**prtMarkerColorantIndex**) in the Printer
 2357 MIB[print-mib]
 2358 AND/OR
 2359 OCTETS: MULTI-ROW: the name of the colorant requested.
 2360

2361 NOTE - The name (**JmJobStringTC**) values correspond to the name values of the
 2362 **prtMarkerColorantValue** object in the Printer MIB. Examples are: red, blue.
 2363

2364 **colorantConsumed(173), Integer32(-2..2147483647)**
 2365 **AND/OR**
 2366 **JmJobStringTC(SIZE(0..63))**
 2367 INTEGER: MULTI-ROW: The index (**prtMarkerColorantIndex**) in the Printer
 2368 MIB[print-mib]
 2369 AND/OR
 2370 OCTETS: MULTI-ROW: the name of the colorant consumed.
 2371

2372 NOTE - The name (**JmJobStringTC**) values correspond to the name values of the
 2373 **prtMarkerColorantValue** object in the Printer MIB. Examples are: red, blue
 2374
 2375

2376 ++++++
 2377 + **Time attributes (set by server or device)**
 2378 +
 2379 + **This section of attributes are ones that are set by the**
 2380 + **server or device that accepts jobs. Two forms of time are**
 2381 + **provided. Each form is represented in a separate attribute.**
 2382 + **See section 0 and section 0 for the**
 2383 + **conformance requirements for time attribute for agents and**
 2384 + **monitoring applications, respectively. The two forms are:**
 2385 +
 2386 + **'DateAndTime' is an 8 or 11 octet binary encoded year,**
 2387 + **month, day, hour, minute, second, deci-second with**
 2388 + **optional offset from UTC. See SNMPv2-TC [SMIV2-TC].**
 2389 +
 2390 + **NOTE: 'DateAndTime' is not printable characters; it is**
 2391 + **binary.**
 2392 +
 2393 + **'JmTimeStampTC' is the time of day measured in the number of**
 2394 + **seconds since the system was booted.**

```

2395      ++++++
2396
2397      jobSubmissionToServerTime(190),           JmTimeStampTC
2398                                           AND/OR
2399                                           DateAndTime
2400      INTEGER: Configuration 3 only: The time
2401      AND/OR
2402      OCTETS: the date and time that the job was submitted to the server (as distinguished
2403      from the device which uses jobSubmissionTime).
2404
2405      jobSubmissionTime(191),                   JmTimeStampTC
2406                                           AND/OR
2407                                           DateAndTime
2408      INTEGER: Configurations 1, 2, and 3: The time
2409      AND/OR
2410      OCTETS: the date and time that the job was submitted to the server or device to which
2411      the agent is providing access.
2412
2413
2414      jobStartedBeingHeldTime(192),             JmTimeStampTC
2415                                           AND/OR
2416                                           DateAndTime
2417
2418      INTEGER: The time
2419      AND/OR
2420      OCTETS: the date and time that the job last entered the pendingHeld state. If the job
2421      has never entered the pendingHeld state, then the value SHALL be '0' or the attribute
2422      SHALL not be present in the table.
2423
2424      jobStartedProcessingTime(193),           JmTimeStampTC
2425                                           AND/OR
2426                                           DateAndTime
2427
2428      INTEGER: The time
2429      AND/OR
2430      OCTETS: the date and time that the job started processing.
2431
2432      jobCompletionTime(194),                   JmTimeStampTC
2433                                           AND/OR
2434                                           DateAndTime
2435
2436      INTEGER: The time
2437      AND/OR
2438      OCTETS: the date and time that the job entered the completed, canceled, or aborted
2439      state.
2440
2441      jobProcessingCPUtime(195)                 Integer32(-2..2147483647)
2442      UNITS 'seconds'
2443      INTEGER: The amount of CPU time in seconds that the job has been in the processing
2444      state. If the job enters the processingStopped state, that elapsed time SHALL not be

```


2443 included. In other words, the **jobProcessingCPUTime** value SHOULD be relatively
2444 repeatable when the same job is processed again on the same device."
2445

2446 REFERENCE

2447 "See Section 0 entitled 'The Attribute Mechanism' for a description of this textual-convention
2448 and its use in the **jmAttributeTable**.

2449 This is a type 2 enumeration. See Section 0."
2450

2451 SYNTAX INTEGER {

2452 other(1),
2453 unknown(2),
2454 jobStateReasons2(3),
2455 jobStateReasons3(4),
2456 jobStateReasons4(5),
2457 processingMessage(6),
2458 processingMessageNaturalLanguageTag(7),
2459 jobCodedCharSet(8),
2460 jobNaturalLanguageTag(9),

2461 jobURI(20),
2462 jobAccountName(21),
2463 serverAssignedJobName(22),
2464 jobName(23),
2465 jobServiceTypes(24),
2466 jobSourceChannelIndex(25),
2467 jobSourcePlatformType(26),
2468 submittingServerName(27),
2469 submittingApplicationName(28),
2470 jobOriginatingHost(29),
2471 deviceNameRequested(30),
2472 queueNameRequested(31),
2473 physicalDevice(32),
2474 numberOfDocuments(33),
2475 fileName(34),
2476 documentName(35),
2477 jobComment(36),
2478 documentFormatIndex(37),
2479 documentFormat(38),

2481 jobPriority(50),
2482 jobProcessAfterDateAndTime(51),
2483 jobHold(52),
2484 jobHoldUntil(53),
2485 outputBin(54),
2486 sides(55),
2487 finishing(56),

2489 printQualityRequested(70),
2490 printQualityUsed(71),
2491

```
2492         printerResolutionRequested(72),
2493         printerResolutionUsed(73),
2494         tonerEcomonyRequested(74),
2495         tonerEcomonyUsed(75),
2496         tonerDensityRequested(76),
2497         tonerDensityUsed(77),
2498
2499         jobCopiesRequested(90),
2500         jobCopiesCompleted(91),
2501         documentCopiesRequested(92),
2502         documentCopiesCompleted(93),
2503         jobKOctetsTransferred(94),
2504         sheetCompletedCopyNumber(95),
2505         sheetCompletedDocumentNumber(96),
2506         jobCollationType(97),
2507
2508         impressionsSpooled(110),
2509         impressionsSentToDevice(111),
2510         impressionsInterpreted(112),
2511         impressionsCompletedCurrentCopy(113),
2512         fullColorImpressionsCompleted(114),
2513         highlightColorImpressionsCompleted(115),
2514
2515         pagesRequested(130),
2516         pagesCompleted(131),
2517         pagesCompletedCurrentCopy(132),
2518
2519         sheetsRequested(150),
2520         sheetsCompleted(151),
2521         sheetsCompletedCurrentCopy(152),
2522
2523         mediumRequested(170),
2524         mediumConsumed(171),
2525         colorantRequested(172),
2526         colorantConsumed(173),
2527
2528         jobSubmissionToServerTime(190),
2529         jobSubmissionTime(191),
2530         jobStartedBeingHeldTime(192),
2531         jobStartedProcessingTime(193),
2532         jobCompletionTime(194),
2533         jobProcessingCPUTime(195)
2534     }
2535
2536
2537
2538
2539 JmJobServiceTypesTC ::= TEXTUAL-CONVENTION
```

2540 STATUS current
 2541 DESCRIPTION
 2542 "Specifies the type(s) of service to which the job has been submitted (print, fax, scan, etc.). The
 2543 service type is represented as an enum that is bit encoded with each job service type so that
 2544 more general and arbitrary services can be created, such as services with more than one
 2545 destination type, or ones with only a source or only a destination. For example, a job service
 2546 might **scan**, **faxOut**, and **print** a single job. In this case, three bits would be set in the
 2547 **jobServiceTypes** attribute, corresponding to the hexadecimal values: **0x8** + **0x20** + **0x4**,
 2548 respectively, yielding: **0x2C**.
 2549
 2550 Whether this attribute is set from a job attribute supplied by the job submission client or is set
 2551 by the recipient job submission server or device depends on the job submission protocol. With
 2552 either implementation, the agent SHALL return a non-zero value for this attribute indicating the
 2553 type of the job.
 2554
 2555 One of the purposes of this attribute is to permit a requester to filter out jobs that are not of
 2556 interest. For example, a printer operator MAY only be interested in jobs that include printing.
 2557 That is why the attribute is in the job identification category.
 2558
 2559 The following service component types are defined (in hexadecimal) and are assigned a
 2560 separate bit value for use with the **jobServiceTypes** attribute:
 2561
 2562 **other 0x1**
 2563 The job contains some instructions that are not one of the identified types.
 2564
 2565 **unknown 0x2**
 2566 The job contains some instructions whose type is unknown to the agent.
 2567
 2568 **print 0x4**
 2569 The job contains some instructions that specify printing
 2570
 2571 **scan 0x8**
 2572 The job contains some instructions that specify scanning
 2573
 2574 **faxIn 0x10**
 2575 The job contains some instructions that specify receive fax
 2576
 2577 **faxOut 0x20**
 2578 The job contains some instructions that specify sending fax
 2579
 2580 **getFile 0x40**
 2581 The job contains some instructions that specify accessing files or documents
 2582
 2583 **putFile 0x80**
 2584 The job contains some instructions that specify storing files or documents
 2585
 2586 **mailList 0x100**
 2587 The job contains some instructions that specify distribution of documents using an
 2588 electronic mail system."

2589 REFERENCE

2590 "These bit definitions are the equivalent of a type 2 enum except that combinations of them
2591 MAY be used together. See section 0."

2592 SYNTAX INTEGER(0..2147483647) -- 31 bits, all but sign bit

2593

2594

2595

2596

2597 **JmJobStateReasons1TC** ::= TEXTUAL-CONVENTION

2598 STATUS current

2599 DESCRIPTION

2600 "The **JmJobStateReasonsN**TC ($N=1..4$) textual-conventions are used with the
2601 **jmJobStateReasons1** object and **jobStateReasonsN** ($N=2..4$), respectively, to provide
2602 additional information regarding the current **jmJobState** object value. These values MAY be
2603 used with any job state or states for which the reason makes sense.

2604

2605 NOTE - While values cannot be added to the **jmJobState** object without impacting deployed
2606 clients that take actions upon receiving **jmJobState** values, it is the intent that additional
2607 **JmJobStateReasonsN**TC enums can be defined and registered without impacting such
2608 deployed clients. In other words, the **jmJobStateReasons1** object and **jobStateReasonsN**
2609 attributes are intended to be extensible.

2610

2611 NOTE - The Job Monitoring MIB contains a superset of the IPP values[ipp-model] for the IPP
2612 'job-state-reasons' attribute, since the Job Monitoring MIB is intended to cover other job
2613 submission protocols as well. Also some of the names of the reasons have been changed from
2614 'printer' to 'device', since the Job Monitoring MIB is intended to cover additional types of
2615 devices, including input devices, such as scanners.

2616

2617 The following standard values are defined (in hexadecimal) as *powers of two*, since multiple
2618 values MAY be used at the same time. For ease of understanding, the
2619 **JmJobStateReasons1TC** reasons are presented in the order in which the reasons are likely to
2620 occur (if implemented), starting with the '**jobIncoming**' value and ending with the
2621 '**jobCompletedWithErrors**' value.

2622

2623 **other** **0x1**

2624 The job state reason is not one of the standardized or registered reasons.

2625

2626 **unknown** **0x2**

2627 The job state reason is not known to the agent or is indeterminate.

2628

2629 **jobIncoming** **0x4**

2630 The job has been accepted by the server or device, but the server or device is expecting
2631 (1) additional operations from the client to finish creating the job and/or (2) is
2632 accessing/accepting document data.

2633

2634	submissionInterrupted	0x8
2635	The job was not completely submitted for some unforeseen reason, such as: (1) the server	
2636	has crashed before the job was closed by the client, (2) the server or the document	
2637	transfer method has crashed in some non-recoverable way before the document data was	
2638	entirely transferred to the server, (3) the client crashed or failed to close the job before the	
2639	time-out period.	
2640		
2641	jobOutgoing	0x10
2642	Configuration 2 only: The server is transmitting the job to the device.	
2643		
2644	jobHoldSpecified	0x20
2645	The value of the job's jobHold(52) attribute is TRUE. The job SHALL NOT be a	
2646	candidate for processing until this reason is removed and there are no other reasons to	
2647	hold the job.	
2648		
2649	jobHoldUntilSpecified	0x40
2650	The value of the job's jobHoldUntil(53) attribute specifies a time period that is still in the	
2651	future. The job SHALL NOT be a candidate for processing until this reason is removed	
2652	and there are no other reasons to hold the job.	
2653		
2654	jobProcessAfterSpecified	0x80
2655	The value of the job's jobProcessAfterDateAndTime(51) attribute specifies a time that is	
2656	still in the future. The job SHALL NOT be a candidate for processing until this reason is	
2657	removed and there are no other reasons to hold the job.	
2658		
2659	resourcesAreNotReady	0x100
2660	At least one of the resources needed by the job, such as media, fonts, resource objects,	
2661	etc., is not ready on any of the physical devices for which the job is a candidate. This	
2662	condition MAY be detected when the job is accepted, or subsequently while the job is	
2663	pending or processing , depending on implementation.	
2664		
2665	deviceStoppedPartly	0x200
2666	One or more, but not all, of the devices to which the job is assigned are stopped. If all of	
2667	the devices are stopped (or the only device is stopped), the deviceStopped reason	
2668	SHALL be used.	
2669		
2670	deviceStopped	0x400
2671	The device(s) to which the job is assigned is (are all) stopped.	
2672		
2673	jobInterpreting	0x800
2674	The device to which the job is assigned is interpreting the document data.	
2675		
2676	jobPrinting	0x1000
2677	The output device to which the job is assigned is marking media. This attribute is useful	
2678	for servers and output devices which spend a great deal of time processing (1) when no	
2679	marking is happening and then want to show that marking is now happening or (2) when	
2680	the job is in the process of being canceled or aborted while the job remains in the	
2681	processing state, but the marking has not yet stopped so that impression or sheet counts	
2682	are still increasing for the job.	

2683		
2684	jobCanceledByUser	0x2000
2685	The job was canceled by the owner of the job, i.e., by a user whose name is the same as	
2686	the value of the job's jmJobOwner object, or by some other authorized end-user, such as	
2687	a member of the job owner's security group.	
2688		
2689	jobCanceledByOperator	0x4000
2690	The job was canceled by the operator, i.e., by a user who has been authenticated as having	
2691	operator privileges (whether local or remote).	
2692		
2693	jobCanceledAtDevice	0x8000
2694	The job was canceled by an unidentified local user, i.e., a user at a console at the device.	
2695		
2696	abortedBySystem	0x10000
2697	The job (1) is in the process of being aborted, (2) has been aborted by the system and	
2698	placed in the ' aborted ' state, or (3) has been aborted by the system and placed in the	
2699	'pendingHeld' state, so that a user or operator can manually try the job again.	
2700		
2701	processingToStopPoint	0x20000
2702	The requester has issued an operation to cancel or interrupt the job or the server/device	
2703	has aborted the job, but the server/device is still performing some actions on the job until	
2704	a specified stop point occurs or job termination/cleanup is completed.	
2705		
2706	This reason is recommended to be used in conjunction with the processing job state to	
2707	indicate that the server/device is still performing some actions on the job while the job	
2708	remains in the processing state. After all the job's resources consumed counters have	
2709	stopped incrementing, the server/device moves the job from the processing state to the	
2710	canceled or aborted job states.	
2711		
2712	serviceOffLine	0x40000
2713	The service or document transform is off-line and accepting no jobs. All pending jobs	
2714	are put into the pendingHeld state. This situation could be true if the service's or	
2715	document transform's input is impaired or broken.	
2716		
2717	jobCompletedSuccessfully	0x80000
2718	The job completed successfully.	
2719		
2720	jobCompletedWithWarnings	0x100000
2721	The job completed with warnings.	
2722		
2723	jobCompletedWithErrors	0x200000
2724	The job completed with errors (and possibly warnings too).	
2725		
2726		
2727	The following additional job state reasons have been added to represent job states that are in	
2728	ISO DPA[iso-dpa] and other job submission protocols:	
2729		

2778	discardTimeArrived	0x4
2779	The job has been deleted due to the fact that the time specified by the job's job-discard-	
2780	time attribute has arrived.	
2781		
2782	postProcessingFailed	0x8
2783	The post-processing agent failed while trying to log accounting attributes for the job;	
2784	therefore the job has been placed into the completed state with the jobRetained	
2785	jmJobStateReasons1 object value for a system-defined period of time, so the	
2786	administrator can examine it, resubmit it, etc.	
2787		
2788	jobTransforming	0x10
2789	The server/device is interpreting document data and producing another electronic	
2790	representation.	
2791		
2792	maxJobFaultCountExceeded	0x20
2793	The job has faulted several times and has exceeded the administratively defined fault	
2794	count limit.	
2795		
2796	devicesNeedAttentionTimeOut	0x40
2797	One or more document transforms that the job is using needs human intervention in order	
2798	for the job to make progress, but the human intervention did not occur within the site-	
2799	settable time-out value.	
2800		
2801	needsKeyOperatorTimeOut	0x80
2802	One or more devices or document transforms that the job is using need a specially trained	
2803	operator (who may need a key to unlock the device and gain access) in order for the job to	
2804	make progress, but the key operator intervention did not occur within the site-settable	
2805	time-out value.	
2806		
2807	jobStartWaitTimeOut	0x100
2808	The server/device has stopped the job at the beginning of processing to await human	
2809	action, such as installing a special cartridge or special non-standard media, but the job	
2810	was not resumed within the site-settable time-out value and the server/device has	
2811	transitioned the job to the pendingHeld state.	
2812		
2813	jobEndWaitTimeOut	0x200
2814	The server/device has stopped the job at the end of processing to await human action,	
2815	such as removing a special cartridge or restoring standard media, but the job was not	
2816	resumed within the site-settable time-out value and the server/device has transitioned the	
2817	job to the completed state.	
2818		
2819	jobPasswordWaitTimeOut	0x400
2820	The server/device has stopped the job at the beginning of processing to await input of the	
2821	job's password, but the password was not received within the site-settable time-out value.	
2822		
2823	deviceTimedOut	0x800
2824	A device that the job was using has not responded in a period specified by the device's	
2825	site-settable attribute.	
2826		

2827	connectingToDeviceTimeOut	0x1000
2828	The server is attempting to connect to one or more devices which may be dial-up, polled,	
2829	or queued, and so may be busy with traffic from other systems, but server was unable to	
2830	connect to the device within the site-settable time-out value.	
2831		
2832	transferring	0x2000
2833	The job is being transferred to a down stream server or downstream device.	
2834		
2835	queuedInDevice	0x4000
2836	The server/device has queued the job in a down stream server or downstream device.	
2837		
2838	jobQueued	0x8000
2839	The server/device has queued the document data.	
2840		
2841	jobCleanup	0x10000
2842	The server/device is performing cleanup activity as part of ending normal processing.	
2843		
2844	jobPasswordWait	0x20000
2845	The server/device has selected the job to be next to process, but instead of assigning	
2846	resources and starting the job processing, the server/device has transitioned the job to the	
2847	pendingHeld state to await entry of a password (and dispatched another job, if there is	
2848	one).	
2849		
2850	validating	0x40000
2851	The server/device is validating the job <i>after</i> accepting the job.	
2852		
2853	queueHeld	0x80000
2854	The operator has held the entire job set or queue.	
2855		
2856	jobProofWait	0x100000
2857	The job has produced a single proof copy and is in the pendingHeld state waiting for the	
2858	requester to issue an operation to release the job to print normally, obeying any job and	
2859	document copy attributes that were originally submitted.	
2860		
2861	heldForDiagnostics	0x200000
2862	The system is running intrusive diagnostics, so that all jobs are being held.	
2863		
2864	noSpaceOnServer	0x800000
2865	There is no room on the server to store all of the job.	
2866		
2867	pinRequired	0x1000000
2868	The System Administrator settable device policy is (1) to require PINs, and (2) to hold	
2869	jobs that do not have a pin supplied as an input parameter when the job was created.	
2870		
2871	exceededAccountLimit	0x2000000
2872	The account for which this job is drawn has exceeded its limit. This condition SHOULD	
2873	be detected before the job is scheduled so that the user does not wait until his/her job is	
2874	scheduled only to find that the account is overdrawn. This condition MAY also occur	
2875	while the job is processing either as processing begins or part way through processing.	

2925 processing. The server or device is keeping the job in the **pendingHeld** state until an
2926 operator can determine what to do with the job."

2927 REFERENCE

2928 "These bit definitions are the equivalent of a type 2 enum except that combinations of them
2929 may be used together. See section 0. The remaining bits are reserved for future standardization
2930 and/or registration. See the description under **JmJobStateReasons1TC** and the
2931 **jobStateReasons3** attribute."

2932 SYNTAX **INTEGER(0..2147483647)** -- 31 bits, all but sign bit

2933

2934

2935

2936

2937

2938 **JmJobStateReasons4TC** ::= TEXTUAL-CONVENTION

2939 STATUS current

2940 DESCRIPTION

2941 "This textual-convention is used in the **jobStateReasons4** attribute to provides additional
2942 information regarding the **jmJobState** object. See the description under
2943 **JmJobStateReasons1TC** for additional information that applies to all reasons.

2944

2945 The following standard values are defined (in hexadecimal) as *powers of two*, since multiple
2946 values may be used at the same time:

2947

2948 none yet defined. These bits are reserved for future standardization and/or registration."

2949 REFERENCE

2950 "These bit definitions are the equivalent of a type 2 enum except that combinations of them
2951 may be used together. See section 0. See the description under **JmJobStateReasons1TC** and
2952 the **jobStateReasons4** attribute."

2953

2954 SYNTAX **INTEGER(0..2147483647)** -- 31 bits, all but sign bit

```

2955
2956 jobmonMIBObjects OBJECT IDENTIFIER ::= { jobmonMIB 1 }
2957
2958 -- The General Group (MANDATORY)
2959
2960 -- The jmGeneralGroup consists entirely of the jmGeneralTable.
2961
2962 jmGeneral OBJECT IDENTIFIER ::= { jobmonMIBObjects 1 }
2963
2964 jmGeneralTable OBJECT-TYPE
2965     SYNTAX      SEQUENCE OF JmGeneralEntry
2966     MAX-ACCESS  not-accessible
2967     STATUS      current
2968     DESCRIPTION
2969         "The jmGeneralTable consists of information of a general nature that are per-job-set, but are
2970         not per-job. See Section 0 entitled 'Terminology and Job Model' for the definition of a job set."
2971     REFERENCE
2972         "The MANDATORY-GROUP macro specifies that this group is MANDATORY."
2973     ::= { jmGeneral 1 }
2974
2975 jmGeneralEntry OBJECT-TYPE
2976     SYNTAX      JmGeneralEntry
2977     MAX-ACCESS  not-accessible
2978     STATUS      current
2979     DESCRIPTION
2980         "Information about a job set (queue).
2981
2982         An entry SHALL exist in this table for each job set."
2983     INDEX { jmGeneralJobSetIndex }
2984     ::= { jmGeneralTable 1 }
2985
2986 JmGeneralEntry ::= SEQUENCE {
2987     jmGeneralJobSetIndex           Integer32(1..32767),
2988     jmGeneralNumberOfActiveJobs   Integer32(0..2147483647),
2989     jmGeneralOldestActiveJobIndex Integer32(0..2147483647),
2990     jmGeneralNewestActiveJobIndex Integer32(0..2147483647),
2991     jmGeneralJobPersistence       Integer32(15..2147483647),
2992     jmGeneralAttributePersistence Integer32(15..2147483647),
2993     jmGeneralJobSetName          JmUTF8StringTC(SIZE(0..63))
2994 }
2995
2996 jmGeneralJobSetIndex OBJECT-TYPE
2997     SYNTAX      Integer32(1..32767)
2998     MAX-ACCESS  not-accessible
2999     STATUS      current
3000     DESCRIPTION
3001         "A unique value for each job set in this MIB. The jmJobTable and jmAttributeTable tables
3002         have this same index as their primary index.
3003

```

3004 The value(s) of the **jmGeneralJobSetIndex** SHALL be persistent across power cycles, so that
3005 clients that have retained **jmGeneralJobSetIndex** values will access the same job sets upon
3006 subsequent power-up.
3007

3008 An implementation that has only one job set, such as a printer with a single queue, SHALL hard
3009 code this object with the value **1**."

3010 REFERENCE
3011 "See Section 0 entitled 'Terminology and Job Model' for the definition of a job set.
3012 Corresponds to the first index in **jmJobTable** and **jmAttributeTable**."
3013 ::= { jmGeneralEntry 1 }
3014

3015 **jmGeneralNumberOfActiveJobs** OBJECT-TYPE
3016 SYNTAX Integer32(0..2147483647)
3017 MAX-ACCESS read-only
3018 STATUS current
3019 DESCRIPTION
3020 "The current number of 'active' jobs in the **jmJobIDTable**, **jmJobTable**, and
3021 **jmAttributeTable**, i.e., the total number of jobs that are in the **pending**, **processing**, or
3022 **processingStopped** states. See the **JmJobStateTC** textual-convention for the exact
3023 specification of the semantics of the job states."
3024 DEFVAL { 0 } -- no jobs
3025 ::= { jmGeneralEntry 2 }
3026

3027 **jmGeneralOldestActiveJobIndex** OBJECT-TYPE
3028 SYNTAX Integer32 (0..2147483647)
3029 MAX-ACCESS read-only
3030 STATUS current
3031 DESCRIPTION
3032 "The **jmJobIndex** of the oldest job that is still in one of the 'active' states **pending**,
3033 **processing**, or **processingStopped**). In other words, the index of the 'active' job that has been
3034 in the job tables the longest.
3035
3036 If there are no active jobs, the agent SHALL set the value of this object to **0**."
3037 REFERENCE
3038 "See Section 0 entitled 'The Job Tables and the Oldest Active and Newest Active Indexes' for a
3039 description of the usage of this object."
3040 DEFVAL { 0 } -- no active jobs
3041 ::= { jmGeneralEntry 3 }
3042

3043 **jmGeneralNewestActiveJobIndex** OBJECT-TYPE
3044 SYNTAX Integer32 (0..2147483647)
3045 MAX-ACCESS read-only
3046 STATUS current
3047 DESCRIPTION
3048 "The **jmJobIndex** of the newest job that is in one of the 'active' states **pending**, **processing**, or
3049 **processingStopped**). In other words, the index of the 'active' job that has been most recently
3050 added to the **job tables**.
3051

3052 When all jobs become 'inactive', i.e., enter the **pendingHeld**, **completed**, **canceled**, or **aborted**
3053 states, the agent SHALL set the value of this object to **0**."

3054 REFERENCE

3055 "See Section 0 entitled 'The Job Tables and the Oldest Active and Newest Active Indexes' for a
3056 description of the usage of this object."

3057 DEFVAL { 0 } -- no active jobs

3058 ::= { jmGeneralEntry 4 }

3059

3060 **jmGeneralJobPersistence** OBJECT-TYPE

3061 SYNTAX **Integer32(15..2147483647)**

3062 UNITS "seconds"

3063 MAX-ACCESS read-only

3064 STATUS current

3065 DESCRIPTION

3066 "The minimum time in seconds for this instance of the Job Set that an entry SHALL remain in
3067 the **jmJobIDTable** and **jmJobTable** after **processing** has *completed*, i.e., the minimum time in
3068 seconds starting when the job enters the **completed**, **canceled**, or **aborted** state.

3069

3070 Configuring this object is implementation-dependent.

3071

3072 This value SHALL be equal to or greater than the value of **jmGeneralAttributePersistence**.
3073 This value SHOULD be at least 60 which gives a monitoring application one minute in which
3074 to poll for job data."

3075 DEFVAL { 60 } -- one minute

3076 ::= { jmGeneralEntry 5 }

3077

3078 **jmGeneralAttributePersistence** OBJECT-TYPE

3079 SYNTAX **Integer32(15..2147483647)**

3080 UNITS "seconds"

3081 MAX-ACCESS read-only

3082 STATUS current

3083 DESCRIPTION

3084 "The minimum time in seconds for this instance of the Job Set that an entry SHALL remain in
3085 the **jmAttributeTable** after **processing** has *completed*, i.e., the time in seconds starting when
3086 the job enters the **completed**, **canceled**, or **aborted** state.

3087

3088 Configuring this object is implementation-dependent.

3089

3090 This value SHOULD be at least 60 which gives a monitoring application one minute in which
3091 to poll for job data."

3092 DEFVAL { 60 } -- one minute

3093 ::= { jmGeneralEntry 6 }

3094

3095 **jmGeneralJobSetName** OBJECT-TYPE

3096 SYNTAX **JmUTF8StringTC(SIZE(0..63))**

3097 MAX-ACCESS read-only

3098 STATUS current

3099 DESCRIPTION

3100 "The human readable name of this job set assigned by the system administrator (by means
3101 outside of this MIB). Typically, this name SHOULD be the name of the job queue. If a server
3102 or device has only a single job set, this object can be the administratively assigned name of the
3103 server or device itself. This name does not need to be unique, though each job set in a single
3104 Job Monitoring MIB SHOULD have distinct names.
3105

3106 NOTE - If the job set corresponds to a single printer and the Printer MIB is implemented, this
3107 value SHOULD be the same as the **prtGeneralPrinterName** object in the draft Printer MIB. If
3108 the job set corresponds to an IPP Printer, this value SHOULD be the same as the IPP 'printer-
3109 name' Printer attribute.
3110

3111 NOTE - The purpose of this object is to help the user of the job monitoring application
3112 distinguish between several job sets in implementations that support more than one job set."
3113

3114 REFERENCE
3115 "See the OBJECT compliance macro for the minimum maximum length required for
3116 conformance."
3117 DEFVAL { 'H } -- empty string
3118 ::= { jmGeneralEntry 7 }
3119
3120
3121
3122

3123 -- The Job ID Group (MANDATORY)
3124

3125 -- The **jmJobIDGroup** consists entirely of the **jmJobIDTable**.
3126

3127 jmJobID OBJECT IDENTIFIER ::= { jobmonMIBObjects 2 }
3128

3129 jmJobIDTable OBJECT-TYPE
3130 SYNTAX SEQUENCE OF JmJobIDEntry
3131 MAX-ACCESS not-accessible
3132 STATUS current
3133 DESCRIPTION
3134 "The **jmJobIDTable** provides a correspondence map (1) between the job submission ID that a
3135 client uses to refer to a job and (2) the **jmGeneralJobSetIndex** and **jmJobIndex** that the Job
3136 Monitoring MIB agent assigned to the job and that are used to access the job in all of the other
3137 tables in the MIB. If a monitoring application already knows the **jmGeneralJobSetIndex** and
3138 the **jmJobIndex** of the job it is querying, that application NEED NOT use the **jmJobIDTable**."
3139

3140 REFERENCE
3141 "The MANDATORY-GROUP macro specifies that this group is MANDATORY."
3142 ::= { jmJobID 1 }
3143

3144 jmJobIDEntry OBJECT-TYPE
3145 SYNTAX JmJobIDEntry
3146 MAX-ACCESS not-accessible
3147 STATUS current
3148 DESCRIPTION

3148 "The map from (1) the **jmJobSubmissionID** to (2) the **jmGeneralJobSetIndex** and
 3149 **jmJobIndex**.

3150

3151 An entry SHALL exist in this table for each job currently known to the agent for all job sets and
 3152 job states. There MAY be more than one **jmJobIDEntry** that maps to a single job. This many
 3153 to one mapping can occur when more than one network entity along the job submission path
 3154 supplies a job submission ID. See Section 0. However, each job SHALL appear once and in
 3155 one and only one job set."

3156 INDEX { **jmJobSubmissionID** }

3157 ::= { jmJobIDTable 1 }

3158

3159 JmJobIDEntry ::= SEQUENCE {

3160 **jmJobSubmissionID** OCTET STRING(SIZE(48)),

3161 **jmJobIDJobSetIndex** Integer32(0..32767),

3162 **jmJobIDJobIndex** Integer32(0..2147483647)

3163 }

3164

3165 **jmJobSubmissionID** OBJECT-TYPE

3166 SYNTAX OCTET STRING(SIZE(48))

3167 MAX-ACCESS not-accessible

3168 STATUS current

3169 DESCRIPTION

3170 "A quasi-unique 48-octet fixed-length string ID which identifies the job within a particular
 3171 client-server environment. There are multiple formats for the **jmJobSubmissionID**. Each
 3172 format SHALL be uniquely identified. See the **JmJobSubmissionIDTypeTC** textual
 3173 convention. Each format SHALL be registered using the procedures of a type 2 enum. See
 3174 section 0 entitled: 'PWG Registration of Job Submission Id Formats'.

3175

3176 If the requester (client or server) does not supply a job submission ID in the job submission
 3177 protocol, then the recipient (server or device) SHALL assign a job submission ID using any of
 3178 the standard formats that have been reserved for agents and adding the final 8 octets to
 3179 distinguish the ID from others submitted from the same requester.

3180

3181 The monitoring application, whether in the client or running separately, MAY use the job
 3182 submission ID to help identify which **jmJobIndex** was assigned by the agent, i.e., in which row
 3183 the job information is in the other tables.

3184

3185 NOTE - fixed-length is used so that a management application can use a shortened GetNext
 3186 varbind (in SNMPv1 and SNMPv2) in order to get the next submission ID, disregarding the
 3187 remainder of the ID in order to access jobs independent of the trailing identifier part, e.g., to get
 3188 all jobs submitted by a particular **jmJobOwner** or submitted from a particular MAC address."

3189 REFERENCE

3190 "See the **JmJobSubmissionIDTypeTC** textual convention.
 3191 See APPENDIX B - Support of Job Submission Protocols."

3192 ::= { jmJobIDEntry 1 }

3193

3194 **jmJobIDJobSetIndex** OBJECT-TYPE

3195 SYNTAX Integer32(0..32767)

3196 MAX-ACCESS read-only

3197 STATUS current
3198 DESCRIPTION
3199 "This object contains the value of the **jmGeneralJobSetIndex** for the job with the
3200 **jmJobSubmissionID** value, i.e., the job set index of the job set in which the job was placed
3201 when that server or device accepted the job. This 16-bit value in combination with the
3202 **jmJobIDJobIndex** value permits the management application to access the other tables to
3203 obtain the job-specific objects for this job."
3204 REFERENCE
3205 "See **jmGeneralJobSetIndex** in the **jmGeneralTable**."
3206 DEFVAL { 0 } -- 0 indicates no job set index
3207 ::= { jmJobIDEntry 2 }
3208
3209 **jmJobIDJobIndex** OBJECT-TYPE
3210 SYNTAX Integer32(0..2147483647)
3211 MAX-ACCESS read-only
3212 STATUS current
3213 DESCRIPTION
3214 "This object contains the value of the **jmJobIndex** for the job with the **jmJobSubmissionID**
3215 value, i.e., the job index for the job when the server or device accepted the job. This value, in
3216 combination with the **jmJobIDJobSetIndex** value, permits the management application to
3217 access the other tables to obtain the job-specific objects for this job."
3218 REFERENCE
3219 "See **jmJobIndex** in the **jmJobTable**."
3220 DEFVAL { 0 } -- 0 indicates no jmJobIndex value.
3221 ::= { jmJobIDEntry 3 }
3222
3223
3224
3225
3226 -- The Job Group (MANDATORY)
3227
3228 -- The **jmJobGroup** consists entirely of the **jmJobTable**.
3229
3230 jmJob OBJECT IDENTIFIER ::= { jobmonMIBObjects 3 }
3231
3232 jmJobTable OBJECT-TYPE
3233 SYNTAX SEQUENCE OF JmJobEntry
3234 MAX-ACCESS not-accessible
3235 STATUS current
3236 DESCRIPTION
3237 "The **jmJobTable** consists of basic job state and status information for each job in a job set that
3238 (1) monitoring applications need to be able to access in a single SNMP Get operation, (2) that
3239 have a single value per job, and (3) that SHALL always be implemented."
3240 REFERENCE
3241 "The MANDATORY-GROUP macro specifies that this group is MANDATORY."
3242 ::= { jmJob 1 }
3243
3244 jmJobEntry OBJECT-TYPE
3245 SYNTAX JmJobEntry

3246 MAX-ACCESS not-accessible
 3247 STATUS current
 3248 DESCRIPTION
 3249 "Basic per-job state and status information.
 3250
 3251 An entry SHALL exist in this table for each job, no matter what the state of the job is. Each job
 3252 SHALL appear in one and only one job set."
 3253 REFERENCE
 3254 "See Section 0 entitled 'The Job Tables'."
 3255 INDEX { **jmGeneralJobSetIndex**, **jmJobIndex** }
 3256 ::= { jmJobTable 1 }
 3257
 3258 JmJobEntry ::= SEQUENCE {
 3259 **jmJobIndex** Integer32(1..2147483647),
 3260 **jmJobState** JmJobStateTC,
 3261 **jmJobStateReasons1** JmJobStateReasons1TC,
 3262 **jmNumberOfInterveningJobs** Integer32(-2..2147483647),
 3263 **jmJobKOctetsPerCopyRequested** Integer32(-2..2147483647),
 3264 **jmJobKOctetsProcessed** Integer32(-2..2147483647),
 3265 **jmJobImpressionsPerCopyRequested** Integer32(-2..2147483647),
 3266 **jmJobImpressionsCompleted** Integer32(-2..2147483647),
 3267 **jmJobOwner** JmJobStringTC(SIZE(0..63))
 3268 }
 3269
 3270 **jmJobIndex** OBJECT-TYPE
 3271 SYNTAX Integer32(1..2147483647)
 3272 MAX-ACCESS not-accessible
 3273 STATUS current
 3274 DESCRIPTION
 3275 "The sequential, monotonically increasing identifier index for the job generated by the server or
 3276 device when that server or device accepted the job. This index value permits the management
 3277 application to access the other tables to obtain the job-specific row entries."
 3278 REFERENCE
 3279 "See Section 0 entitled 'The Job Tables and the Oldest Active and Newest Active Indexes'.
 3280 See Section 0 entitled 'Job Identification'.
 3281 See also **jmGeneralNewestActiveJobIndex** for the largest value of **jmJobIndex**.
 3282 See **JmJobSubmissionIDTypeTC** for a limit on the size of this index if the agent represents it
 3283 as an 8-digit decimal number."
 3284 ::= { jmJobEntry 1 }
 3285
 3286 **jmJobState** OBJECT-TYPE
 3287 SYNTAX JmJobStateTC
 3288 MAX-ACCESS read-only
 3289 STATUS current
 3290 DESCRIPTION
 3291 "The current state of the job (**pending**, **processing**, **completed**, etc.). Agents SHALL
 3292 implement only those states which are appropriate for the particular implementation. However,
 3293 management applications SHALL be prepared to receive all the standard job states.
 3294

3295 The final value for this object SHALL be one of: **completed**, **canceled**, or **aborted**. The
3296 minimum length of time that the agent SHALL maintain MIB data for a job in the **completed**,
3297 **canceled**, or **aborted** state before removing the job data from the **jmJobIDTable** and
3298 **jmJobTable** is specified by the value of the **jmGeneralJobPersistence** object."
3299 DEFVAL { unknown } -- default is unknown
3300 ::= { jmJobEntry 2 }

3301

3302 **jmJobStateReasons1** OBJECT-TYPE
3303 SYNTAX **JmJobStateReasons1TC**
3304 MAX-ACCESS read-only
3305 STATUS current
3306 DESCRIPTION
3307 "Additional information about the job's current state, i.e., information that augments the value
3308 of the job's **jmJobState** object."
3309

3310 Implementation of any reason values is OPTIONAL, but an agent SHOULD return any reason
3311 information available. These values MAY be used with any job state or states for which the
3312 reason makes sense. Since the Job State Reasons will be more dynamic than the Job State, it is
3313 recommended that a job monitoring application read this object every time **jmJobState** is read.
3314 When the agent cannot provide a reason for the current state of the job, the value of the
3315 **jmJobStateReasons1** object and **jobStateReasonsN** attributes SHALL be **0**."

3316 REFERENCE
3317 "The **jobStateReasonsN** ($N=2..4$) attributes provide further additional information about the
3318 job's current state."
3319 DEFVAL { 0 } -- no reasons
3320 ::= { jmJobEntry 3 }

3321

3322 **jmNumberOfInterveningJobs** OBJECT-TYPE
3323 SYNTAX **Integer32(-2..2147483647)**
3324 MAX-ACCESS read-only
3325 STATUS current
3326 DESCRIPTION
3327 "The number of jobs that are expected to complete processing *before* this job has completed
3328 processing according to the implementation's queuing algorithm, if no other jobs were to be
3329 submitted. In other words, this value is the job's queue position. The agent SHALL return a
3330 value of **0** for this attribute when the job is the next job to complete processing (or has
3331 completed processing)."
3332 DEFVAL { 0 } -- default is no intervening jobs.
3333 ::= { jmJobEntry 4 }

3334

3335 **jmJobKOctetsPerCopyRequested** OBJECT-TYPE
3336 SYNTAX **Integer32(-2..2147483647)**
3337 MAX-ACCESS read-only
3338 STATUS current
3339 DESCRIPTION
3340 "The total size in K (1024) octets of the document(s) being requested to be processed in the job.
3341 The agent SHALL round the actual number of octets up to the next highest K. Thus 0 octets
3342 SHALL be represented as **0'**, 1-1024 octets SHALL be represented as **1'**, 1025-2048 SHALL
3343 be represented as **2'**, etc."

3344
 3345 In computing this value, the server/device SHALL *not* include the multiplicative factors
 3346 contributed by (1) the number of document copies, and (2) the number of job copies,
 3347 independent of whether the device can process multiple copies of the job or document without
 3348 making multiple passes over the job or document data and independent of whether the output is
 3349 collated or not. Thus the server/device computation is independent of the implementation and
 3350 indicates the size of the document(s) measured in K octets independent of the number of
 3351 copies."

3352 DEFVAL { -2 } -- the default is unknown(-2)
 3353 ::= { jmJobEntry 5 }

3354 **jmJobKOctetsProcessed** OBJECT-TYPE

3355 SYNTAX **Integer32(-2..2147483647)**

3356 MAX-ACCESS read-only

3357 STATUS current

3358 DESCRIPTION

3359 "The total number of octets processed by the server or device measured in units of K (1024)
 3360 octets so far. The agent SHALL round the actual number of octets processed up to the next
 3361 higher K. Thus 0 octets SHALL be represented as **0**, 1-1024 octets SHALL be represented as
 3362 **1**, 1025-2048 octets SHALL be **2**, etc. For printing devices, this value is the number
 3363 interpreted by the page description language interpreter rather than what has been marked on
 3364 media.
 3365

3366
 3367 For implementations where multiple copies are produced by the interpreter with only a single
 3368 pass over the data, the final value SHALL be equal to the value of the
 3369 **jmJobKOctetsPerCopyRequested** object. For implementations where multiple copies are
 3370 produced by the interpreter by processing the data for each copy, the final value SHALL be a
 3371 multiple of the value of the **jmJobKOctetsPerCopyRequested** object.
 3372

3373 NOTE - See the **impressionsCompletedCurrentCopy** and **pagesCompletedCurrentCopy**
 3374 attributes for attributes that are reset on each document copy.
 3375

3376 NOTE - The **jmJobKOctetsProcessed** object can be used with the
 3377 **jmJobKOctetsPerCopyRequested** object to provide an indication of the relative progress of
 3378 the job, provided that the multiplicative factor is taken into account for some implementations
 3379 of multiple copies."

3380 DEFVAL { 0 } -- default is no octets processed.
 3381 ::= { jmJobEntry 6 }

3382 **jmJobImpressionsPerCopyRequested** OBJECT-TYPE

3383 SYNTAX **Integer32(-2..2147483647)**

3384 MAX-ACCESS read-only

3385 STATUS current

3386 DESCRIPTION

3387 "The total size in number of impressions of the document(s) submitted.
 3388

3389
 3390 In computing this value, the server/device SHALL *not* include the multiplicative factors
 3391 contributed by (1) the number of document copies, and (2) the number of job copies,
 3392 independent of whether the device can process multiple copies of the job or document without

3393 making multiple passes over the job or document data and independent of whether the output is
3394 collated or not. Thus the server/device computation is independent of the implementation and
3395 reflects the size of the document(s) measured in impressions independent of the number of
3396 copies."
3397 REFERENCE
3398 "See the definition of the term 'impression' in Section0."
3399 DEFVAL { -2 } -- default is unknown(-2)
3400 ::= { jmJobEntry 7 }
3401
3402 **jmJobImpressionsCompleted** OBJECT-TYPE
3403 SYNTAX Integer32(-2..2147483647)
3404 MAX-ACCESS read-only
3405 STATUS current
3406 DESCRIPTION
3407 "The total number of impressions completed for this job so far. For printing devices, the
3408 impressions completed includes interpreting, marking, and stacking the output. For other types
3409 of job services, the number of impressions completed includes the number of impressions
3410 processed.
3411
3412 NOTE - See the **impressionsCompletedCurrentCopy** and **pagesCompletedCurrentCopy**
3413 attributes for attributes that are reset on each document copy.
3414
3415 NOTE - The **jmJobImpressionsCompleted** object can be used with the
3416 **jmJobImpressionsPerCopyRequested** object to provide an indication of the relative progress
3417 of the job, provided that the multiplicative factor is taken into account for some
3418 implementations of multiple copies."
3419 REFERENCE
3420 "See the definition of the term 'impression' in Section0 and the counting example in Section 0
3421 entitled 'Monitoring Job Progress'.
3422 DEFVAL { 0 } -- default is no octets
3423 ::= { jmJobEntry 8 }
3424
3425 **jmJobOwner** OBJECT-TYPE
3426 SYNTAX JmJobStringTC(SIZE(0..63))
3427 MAX-ACCESS read-only
3428 STATUS current
3429 DESCRIPTION
3430 "The coded character set name of the user that submitted the job. The method of assigning this
3431 user name will be system and/or site specific but the method MUST insure that the name is
3432 unique to the network that is visible to the client and target device.
3433
3434 This value SHOULD be the most *authenticated* name of the user submitting the job."
3435 REFERENCE
3436 "See the OBJECT compliance macro for the minimum maximum length required for
3437 conformance."
3438 DEFVAL { ''H } -- empty string
3439 ::= { jmJobEntry 9 }
3440
3441

3442
3443
3444 -- The Attribute Group (MANDATORY)
3445
3446 -- The **jmAttributeGroup** consists entirely of the **jmAttributeTable**.
3447 --
3448 -- Implementation of the two objects in this group is MANDATORY.
3449 -- See Section 0 entitled 'Conformance Considerations'.
3450 -- An agent SHALL implement any attribute if (1) the server or device
3451 -- supports the functionality represented by the attribute and (2) the
3452 -- information is available to the agent.
3453
3454 jmAttribute OBJECT IDENTIFIER ::= { jobmonMIBObjects 4 }
3455
3456 jmAttributeTable OBJECT-TYPE
3457 SYNTAX SEQUENCE OF JmAttributeEntry
3458 MAX-ACCESS not-accessible
3459 STATUS current
3460 DESCRIPTION
3461 "The **jmAttributeTable** SHALL contain attributes of the job and document(s) for each job in a
3462 job set. Instead of allocating distinct objects for each attribute, each attribute is represented as a
3463 separate row in the **jmAttributeTable**."
3464 REFERENCE
3465 "The MANDATORY-GROUP macro specifies that this group is MANDATORY. An agent
3466 SHALL implement any attribute if (1) the server or device supports the functionality
3467 represented by the attribute and (2) the information is available to the agent. "
3468 ::= { **jmAttribute 1** }
3469
3470 jmAttributeEntry OBJECT-TYPE
3471 SYNTAX JmAttributeEntry
3472 MAX-ACCESS not-accessible
3473 STATUS current
3474 DESCRIPTION
3475 "Attributes representing information about the job and document(s) or resources required and/or
3476 consumed.
3477
3478 Each entry in the **jmAttributeTable** is a per-job entry with an extra index for each type of
3479 attribute (**jmAttributeTypeIndex**) that a job can have and an additional index
3480 (**jmAttributeInstanceIndex**) for those attributes that can have multiple instances per job. The
3481 **jmAttributeTypeIndex** object SHALL contain an enum type that indicates the type of attribute
3482 (see the **JmAttributeTypeTC** textual-convention). The value of the attribute SHALL be
3483 represented in either the **jmAttributeValueAsInteger** or **jmAttributeValueAsOctets** objects,
3484 and/or both, as specified in the **JmAttributeTypeTC** textual-convention.
3485
3486 The agent SHALL create rows in the **jmAttributeTable** as the server or device is able to
3487 discover the attributes either from the job submission protocol itself or from the document
3488 PDL. As the documents are interpreted, the interpreter MAY discover additional attributes and
3489 so the agent adds additional rows to this table. As the attributes that represent resources are
3490 actually consumed, the usage counter contained in the **jmAttributeValueAsInteger** object is

3491 incremented according to the units indicated in the description of the **JmAttributeTypeTC**
3492 enum.
3493
3494 The agent SHALL maintain each row in the **jmJobTable** for at least the minimum time after a
3495 job completes as specified by the **jmGeneralAttributePersistence** object.
3496
3497 Zero or more entries SHALL exist in this table for each job in a job set."
3498 REFERENCE
3499 "See Section 0 entitled 'The Attribute Mechanism' for a description of the **jmAttributeTable**."
3500 INDEX { **jmGeneralJobSetIndex**, **jmJobIndex**, **jmAttributeTypeIndex**,
3501 **jmAttributeInstanceIndex** }
3502 ::= { jmAttributeTable 1 }
3503
3504 JmAttributeEntry ::= SEQUENCE {
3505 **jmAttributeTypeIndex** **JmAttributeTypeTC**,
3506 **jmAttributeInstanceIndex** **Integer32(1..32767)**,
3507 **jmAttributeValueAsInteger** **Integer32(-2..2147483647)**,
3508 **jmAttributeValueAsOctets** **OCTET STRING(SIZE(0..63))**
3509 }
3510
3511 **jmAttributeTypeIndex** OBJECT-TYPE
3512 SYNTAX **JmAttributeTypeTC**
3513 MAX-ACCESS not-accessible
3514 STATUS current
3515 DESCRIPTION
3516 "The type of attribute that this row entry represents.
3517
3518 The type MAY identify information about the job or document(s) or MAY identify a resource
3519 required to process the job before the job start processing and/or consumed by the job as the job
3520 is processed.
3521
3522 Examples of job attributes (i.e., apply to the job as a whole) that have only one instance per job
3523 include: **jobCopiesRequested(90)**, **documentCopiesRequested(92)**,
3524 **jobCopiesCompleted(91)**, **documentCopiesCompleted(93)**, while examples of job attributes
3525 that may have more than one instance per job include: **documentFormatIndex(37)**, and
3526 **documentFormat(38)**.
3527
3528 Examples of document attributes (one instance per document) include: **fileName(34)**, and
3529 **documentName(35)**.
3530
3531 Examples of required and consumed resource attributes include: **pagesRequested(130)**,
3532 **mediumRequested(170)**, **pagesCompleted(131)**, and **mediumConsumed(171)**, respectively."
3533 ::= { jmAttributeEntry 1 }
3534
3535 **jmAttributeInstanceIndex** OBJECT-TYPE
3536 SYNTAX **Integer32(1..32767)**
3537 MAX-ACCESS not-accessible
3538 STATUS current
3539 DESCRIPTION

3540 "A running 16-bit index of the attributes of the same type for each job. For those attributes with
 3541 only a single instance per job, this index value SHALL be 1. For those attributes that are a
 3542 single value per document, the index value SHALL be the document number, starting with 1 for
 3543 the first document in the job. Jobs with only a single document SHALL use the index value of
 3544 1. For those attributes that can have multiple values per job or per document, such as
 3545 **documentFormatIndex(37)** or **documentFormat(38)**, the index SHALL be a running index
 3546 for the job as a whole, starting at 1."
 3547 ::= { jmAttributeEntry 2 }

3548 **jmAttributeValueAsInteger** OBJECT-TYPE

3549 SYNTAX **Integer32(-2..2147483647)**

3550 MAX-ACCESS read-only

3551 STATUS current

3552 DESCRIPTION

3553 "The integer value of the attribute. The value of the attribute SHALL be represented as an
 3554 integer if the enum description in the **JmAttributeTypeTC** textual-convention definition has
 3555 the tag: 'INTEGER:'.

3556
 3557 Depending on the enum definition, this object value MAY be an integer, a counter, an index, or
 3558 an enum, depending on the **jmAttributeTypeIndex** value. The units of this value are specified
 3559 in the enum description.

3560
 3561 For those attributes that are accumulating job consumption as the job is processed as specified
 3562 in the **JmAttributeTypeTC** textual-convention, SHALL contain the final value after the job
 3563 completes processing, i.e., this value SHALL indicate the total usage of this resource made by
 3564 the job.

3565
 3566 A monitoring application is able to copy this value to a suitable longer term storage for later
 3567 processing as part of an accounting system.

3568
 3569 Since the agent MAY add attributes representing resources to this table while the job is waiting
 3570 to be processed or being processed, which can be a long time before any of the resources are
 3571 actually used, the agent SHALL set the value of the **jmAttributeValueAsInteger** object to 0
 3572 for resources that the job has not yet consumed.

3573
 3574 Attributes for which the concept of an integer value is meaningless, such as **fileName(34)**,
 3575 **jobName**, and **processingMessage**, do *not* have the 'INTEGER:' tag in the
 3576 **JmAttributeTypeTC** definition and so an agent SHALL always return a value of '-1' to
 3577 indicate 'other' for the value of the **jmAttributeValueAsInteger** object for these attributes.

3578
 3579 For attributes which do have the 'INTEGER:' tag in the **JmAttributeTypeTC** definition, if the
 3580 integer value is not (yet) known, the agent either (1) SHALL not materialize the row in the
 3581 **jmAttributeTable** until the value is known or (2) SHALL return a '-2' to represent an
 3582 'unknown' counting integer value, a '0' to represent an 'unknown' index value, and a '2' to
 3583 represent an 'unknown(2)' enum value."

3584 DEFVAL { -2 } -- default value is unknown(-2)

3585 ::= { jmAttributeEntry 3 }

3586
 3587 **jmAttributeValueAsOctets** OBJECT-TYPE

3589 SYNTAX OCTET STRING(SIZE(0..63))
3590 MAX-ACCESS read-only
3591 STATUS current
3592 DESCRIPTION
3593 "The octet string value of the attribute. The value of the attribute SHALL be represented as an
3594 OCTET STRING if the enum description in the **JmAttributeTypeTC** textual-convention
3595 definition has the tag: 'OCTETS:'.
3596
3597 Depending on the enum definition, this object value MAY be a coded character set string (text),
3598 such as '**JmUTF8StringTC**', or a binary octet string, such as '**DateAndTime**'.
3599
3600 Attributes for which the concept of an octet string value is meaningless, such as
3601 **pagesCompleted**, do *not* have the tag 'OCTETS:' in the **JmAttributeTypeTC** definition and so
3602 the agent SHALL always return a zero length string for the value of the
3603 **jmAttributeValueAsOctets** object.
3604
3605 For attributes which do have the 'OCTETS:' tag in the **JmAttributeTypeTC** definition, if the
3606 OCTET STRING value is not (yet) known, the agent either SHALL not materialize the row in
3607 the **jmAttributeTable** until the value is known or SHALL return a zero-length string."
3608 DEFVAL { ''H } -- empty string
3609 ::= { jmAttributeEntry 4 }
3610

```
3611 -- Notifications and Trapping
3612 -- Reserved for the future
3613
3614 jobmonMIBNotifications OBJECT IDENTIFIER ::= { jobmonMIB 2 }
3615
3616
3617
3618 -- Conformance Information
3619
3620 jmMIBConformance OBJECT IDENTIFIER ::= { jobmonMIB 3 }
3621
3622 -- compliance statements
3623 jmMIBCompliance MODULE-COMPLIANCE
3624     STATUS current
3625     DESCRIPTION
3626         "The compliance statement for agents that implement the
3627         job monitoring MIB."
3628     MODULE -- this module
3629     MANDATORY-GROUPS {
3630         jmGeneralGroup, jmJobIDGroup, jmJobGroup, jmAttributeGroup }
3631
3632     OBJECT jmGeneralJobSetName
3633     SYNTAX JmUTF8StringTC (SIZE(0..8))
3634     DESCRIPTION
3635         "Only 8 octets maximum string length NEED be supported by the agent."
3636
3637     OBJECT jmJobOwner
3638     SYNTAX JmJobStringTC (SIZE(0..16))
3639     DESCRIPTION
3640         "Only 16 octets maximum string length NEED be supported by the agent."
3641
3642 -- There are no CONDITIONALLY MANDATORY or OPTIONAL groups.
3643
3644     ::= { jmMIBConformance 1 }
3645
3646 jmMIBGroups OBJECT IDENTIFIER ::= { jmMIBConformance 2 }
3647
3648 jmGeneralGroup OBJECT-GROUP
3649     OBJECTS {
3650         jmGeneralNumberOfActiveJobs, jmGeneralOldestActiveJobIndex,
3651         jmGeneralNewestActiveJobIndex, jmGeneralJobPersistence,
3652         jmGeneralAttributePersistence, jmGeneralJobSetName }
3653     STATUS current
3654     DESCRIPTION
3655         "The general group."
3656     ::= { jmMIBGroups 1 }
3657
3658 jmJobIDGroup OBJECT-GROUP
3659     OBJECTS {
```

```
3660         jmJobIDJobSetIndex, jmJobIDJobIndex }
3661     STATUS current
3662     DESCRIPTION
3663         "The job ID group."
3664     ::= { jmMIBGroups 2 }
3665
3666     jmJobGroup OBJECT-GROUP
3667         OBJECTS {
3668             jmJobState, jmJobStateReasons1, jmNumberOfInterveningJobs,
3669             jmJobKOctetsPerCopyRequested, jmJobKOctetsProcessed,
3670             jmJobImpressionsPerCopyRequested, jmJobImpressionsCompleted, jmJobOwner }
3671     STATUS current
3672     DESCRIPTION
3673         "The job group."
3674     ::= { jmMIBGroups 3 }
3675
3676     jmAttributeGroup OBJECT-GROUP
3677         OBJECTS {
3678             jmAttributeValueAsInteger, jmAttributeValueAsOctets }
3679     STATUS current
3680     DESCRIPTION
3681         "The attribute group."
3682     ::= { jmMIBGroups 4 }
3683
3684
3685     END
```

3686 5. Appendix A - Implementing the Job Life Cycle

3687 The job object has well-defined states and client operations that affect the transition between the
3688 job states. Internal server and device actions also affect the transitions of the job between the job
3689 states. These states and transitions are referred to as the job's *life cycle*.

3690 Not all implementations of job submission protocols have all of the states of the job model
3691 specified here. The job model specified here is intended to be a superset of most
3692 implementations. It is the purpose of the agent to map the particular implementation's job life
3693 cycle onto the one specified here. The agent MAY omit any states not implemented. Only the
3694 **processing** and **completed** states are required to be implemented by an agent. However, a
3695 conforming management application SHALL be prepared to accept any of the states in the job
3696 life cycle specified here, so that the management application can interoperate with any
3697 conforming agent.

3698 The job states are intended to be user visible. The agent SHALL make these states visible in the
3699 MIB, but only for the subset of job states that the implementation has. Some implementations
3700 MAY need to have sub-states of these user-visible states. The **jmJobStateReasons1** object and
3701 the **jobStateReasonsN** ($N=2..4$) attributes can be used to represent the sub-states of the jobs.

3702 Job states are intended to last a user-visible length of time in most implementations. However,
3703 some jobs may pass through some states in zero time in some situations and/or in some
3704 implementations.

3705 The job model does not specify how accounting and auditing is implemented, except to assume
3706 that accounting and auditing logs are separate from the job life cycle and last longer than job
3707 entries in the MIB. Jobs in the **completed**, **aborted**, or **canceled** states are not logs, since jobs in
3708 these states are accessible via SNMP protocol operations and SHALL be removed from the Job
3709 Monitoring MIB tables after a site-settable or implementation-defined period of time. An
3710 accounting application MAY copy accounting information incrementally to an accounting log as
3711 a job processes, or MAY be copied while the job is in the **canceled**, **aborted**, or **completed**
3712 states, depending on implementation. The same is true for auditing logs.

3713 **The jmJobState object specifies the standard job states. The normal job state transitions**
3714 **are shown in the state transition diagram presented in Table 1.**

3715 6. APPENDIX B - Support of Job Submission Protocols

3716 A companion PWG document, entitled "Job Submission Protocol Mapping
3717 Recommendations for the Job Monitoring MIB" contains the recommended usage of each
3718 of the objects and attributes in this MIB with a number of job submission protocols. In
3719 particular, which job submission ID format should be used is indicated for each job
3720 submission protocol.

3721 Some job submission protocols have support for the client to specify a job submission ID.
3722 A second approach is to enhance the document format to embed the job submission ID in
3723 the document data. This second approach is independent of the job submission protocol.
3724 This appendix lists some examples of these approaches.

3725 Some PJJ implementations wrap a banner page as a PJJ job around a job submitted by a
3726 client. If this results in multiple job submission IDs, the agent SHALL create multiple
3727 **jmJobIDEntry** rows in the **jmJobIDTable** that each point to the same job entry in the
3728 job tables. See the specification of the **jmJobIDEntry**.

3729 7. References

- 3730 [char-set policy] Harald Avelstrand, "IETF Policy on Character Sets and Language",
3731 June 1997. Latest draft: <draft-avelstrand-charset-policy-00.txt>
- 3732 [GB2312] GB 2312-1980, "Chinese People's Republic of China (PRC) mixed one byte
3733 and two byte coded character set"
- 3734 [hr-mib] P. Grillo, S. Waldbusser, "Host Resources MIB", RFC 1514, September 1993
- 3735 [iana] J. Reynolds, and J. Postel, "Assigned Numbers", STD 2, RFC 1700, ISI, October
3736 1994.
- 3737 [IANA-charsets] Coded Character Sets registered by IANA and assigned an enum value
3738 for use in the **CodedCharSet** textual convention imported from the Printer MIB. See
3739 <ftp://ftp.isi.edu/in-notes/iana/assignments/character-sets>
- 3740 [iana-media-types] IANA Registration of MIME media types (MIME content
3741 types/subtypes). See <ftp://ftp.isi.edu/in-notes/iana/assignments/>
- 3742 [ISO-639] ISO 639:1988 (E/F) - Code for Representation of names of languages - The
3743 International Organization for Standardization, 1st edition, 1988.
- 3744 [ISO 646] ISO/IEC 646:1991, "Information technology -- ISO 7-bit coded character set
3745 for information interchange", JTC1/SC2.
- 3746 [ISO 8859] ISO/IEC 8859-1:1987, "Information technology -- 8-bit single byte coded
3747 graphic character sets - Part 1: Latin alphabet No. 1, JTC1/SC2."
- 3748 [ISO 2022] ISO/IEC 2022:1994 - "Information technology -- Character code structure
3749 and extension techniques", JTC1/SC2.
- 3750 [ISO-3166] ISO 3166:1988 (E/F) - Codes for representation of names of countries - The
3751 International Organization for Standardization, 3rd edition, 1988-08-15."
- 3752 [ISO-10646] ISO/IEC 10646-1:1993, "Information technology -- Universal Multiple-
3753 Octet Coded Character Set (UCS) - Part 1: Architecture and Basic Multilingual Plane,
3754 JTC1/SC2.

- 3755 [iso-dpa] ISO/IEC 10175 Document Printing Application (DPA). See
3756 **ftp://ftp.pwg.org/pub/pwg/dpa/**
- 3757 [ipp-model] Internet Printing Protocol (IPP), work in progress on the IETF standards
3758 track. See **draft-ietf-ipp-model-07.txt**. See also **http://www.pwg.org/ipp/index.html**
- 3759 [JIS X0208] JIS X0208-1990, "Japanese two byte coded character set."
- 3760 [mib-II] MIB-II, RFC 1213.
- 3761 [print-mib] The Printer MIB - RFC 1759, proposed IETF standard. Also an Internet-
3762 Draft on the standards track as a draft standard: **draft-ietf-printmib-mib-info-02.txt**
- 3763 [pwg] The Printer Working Group is a printer industry consortium open to any
3764 individuals. For more information, access the PWG web page: <http://www.pwg.org>
- 3765 [req-words] S. Bradner, "Keywords for use in RFCs to Indicate Requirement Levels",
3766 RFC 2119, March 1997.
- 3767 [rfc 1738] Berners-Lee, T., Masinter, L., McCahill, M., "Uniform Resource Locators
3768 (URL)", RFC 1738, December 1994.
- 3769 [RFC-1766] Avelstrand H., "Tags for the Identification of Languages", RFC 1766, March
3770 1995.
- 3771 [rfc 2130] C. Weider, C. Preston, K. Simonsen, H. Alvestrand, R. Atkinson, M. Crispin,
3772 and P. Svanberg, "The Report of the IAB Character Set Workshop held 29 Feb-1 March,
3773 1997", April 1997, RFC 2130.
- 3774 [SMIv2-SMI] J. Case, et al. "Structure of Management Information for Version 2 of the
3775 Simple Network Management Protocol (SNMPv2)", RFC 1902, January 1996.
- 3776 [SMIv2-TC] J. Case, et al. "Textual Conventions for Version 2 of the Simple Network
3777 Management Protocol (SNMPv2)", RFC 1903, January 1996.
- 3778 [tipsi] IEEE 1284.1, Transport-independent Printer System Interface (TIPSI).
- 3779 [URI-spec] Berners-Lee, T., Masinter, L., McCahill, M., "Uniform Resource Locators
3780 (URL)", RFC 1738, December, 1994.
- 3781 [US-ASCII] Coded Character Set - 7-bit American Standard Code for Information
3782 Interchange, ANSI X3.4-1986.
- 3783 [UTF-8] F. Yergeau, "UTF-8, a transformation format of Unicode and ISO 10646", RFC
3784 2044, October 1996.

3785 8. Author's Addresses

3786 Ron Bergman
3787 Dataproducts Corp.

3788 1757 Tapo Canyon Road
3789 Simi Valley, CA 93063-3394

3790
3791 Phone: 805-578-4421
3792 Fax: 805-578-4001
3793 Email: rbergman@dpc.com

3794
3795
3796 Tom Hastings
3797 Xerox Corporation, ESAE-231
3798 701 S. Aviation Blvd.
3799 El Segundo, CA 90245

3800
3801 Phone: 310-333-6413
3802 Fax: 310-333-5514
3803 EMail: hastings@cp10.es.xerox.com

3804
3805
3806 Scott A. Isaacson
3807 Novell, Inc.
3808 122 E 1700 S
3809 Provo, UT 84606
3810
3811 Phone: 801-861-7366
3812 Fax: 801-861-4025
3813 EMail: scott_isaacson@novell.com

3814
3815
3816 Harry Lewis
3817 IBM Corporation
3818 6300 Diagonal Hwy
3819 Boulder, CO 80301
3820
3821 Phone: (303) 924-5337
3822 Fax:
3823 Email: harryl@us.ibm.com

3824
3825
3826 Send comments to the printmib WG using the Job Monitoring Project (JMP)
3827 Mailing List: jmp@pwg.org
3828

3829 To learn how to subscribe, send email to: jmp-request@pwg.org

3830

3831 For further information, access the PWG web page under "JMP":

3832 <http://www.pwg.org/>

3833

3834 Other Participants:

3835 Chuck Adams - Tektronix
3836 Jeff Barnett - IBM
3837 Keith Carter, IBM Corporation
3838 Jeff Copeland - QMS
3839 Andy Davidson - Tektronix
3840 Roger deBry - IBM
3841 Mabry Dozier - QMS
3842 Lee Ferrel - Canon
3843 Steve Gebert - IBM
3844 Robert Herriot - Sun Microsystems Inc.
3845 Shige Kanemitsu - Kyocera
3846 David Kellerman - Northlake Software
3847 Rick Landau - Digital
3848 Harry Lewis - IBM
3849 Pete Loya - HP
3850 Ray Lutz - Cognisys
3851 Jay Martin - Underscore
3852 Mike MacKay, Novell, Inc.
3853 Stan McConnell - Xerox
3854 Carl-Uno Manros, Xerox, Corp.
3855 Pat Nogay - IBM
3856 Bob Pentecost - HP
3857 Rob Rhoads - Intel
3858 David Roach - Unisys
3859 Stuart Rowley - Kyocera
3860 Hiroyuki Sato - Canon
3861 Bob Setterbo - Adobe
3862 Gail Songer, EFI
3863 Mike Timperman - Lexmark
3864 Randy Turner - Sharp
3865 William Wagner - Digital Products
3866 Jim Walker - Dazel
3867 Chris Wellens - Interworking Labs
3868 Rob Whittle - Novell

3869 Don Wright - Lexmark
3870 Lloyd Young - Lexmark
3871 Atsushi Yuki - Kyocera
3872 Peter Zehler, Xerox, Corp.

3873 **9.**

3874 **10. INDEX**

3875 This index includes the textual conventions, the objects, and the attributes. Textual
 3876 conventions all start with the prefix: "**JM**" and end with the suffix: "**TC**". Objects all
 3877 starts with the prefix: "**jm**" followed by the group name. Attributes are identified with
 3878 enums, and so start with any lower case letter and have no special prefix.

3879	colorantConsumed.....	63	3926	JmJobStringTC.....	38
3880	colorantRequested.....	63	3927	jmJobSubmissionID.....	80
3881	deviceNameRequested.....	54	3928	JmJobSubmissionIDTypeTC.....	44
3882	documentCopiesCompleted.....	59	3929	JmMediumTypeTC.....	42
3883	documentCopiesRequested.....	59	3930	JmNaturalLanguageTagTC.....	38
3884	documentFormat.....	56	3931	jmNumberOfInterveningJobs.....	83
3885	documentFormatIndex.....	55	3932	JmPrinterResolutionTC.....	41
3886	documentName.....	55	3933	JmPrintQualityTC.....	41
3887	fileName.....	55	3934	JmTimeStampTC.....	38
3888	finishing.....	57	3935	JmTonerEconomyTC.....	41
3889	fullColorImpressionsCompleted.....	60	3936	JmUTF8StringTC.....	38
3890	highlightColorImpressionsCompleted.....	60	3937	jobAccountName.....	53
3891	impressionsCompletedCurrentCopy.....	60	3938	jobCodedCharSet.....	52
3892	impressionsInterpreted.....	60	3939	jobCollationType.....	60
3893	impressionsSentToDevice.....	60	3940	jobComment.....	55
3894	impressionsSpooled.....	60	3941	jobCompletionTime.....	64
3895	jmAttributeInstanceIndex.....	87	3942	jobCopiesCompleted.....	58
3896	jmAttributeTypeIndex.....	87	3943	jobCopiesRequested.....	58
3897	JmAttributeTypeTC.....	50	3944	jobHold.....	57
3898	jmAttributeValueAsInteger.....	88	3945	jobHoldUntil.....	57
3899	jmAttributeValueAsOctets.....	88	3946	jobKOctetsTransferred.....	59
3900	JmBooleanTC.....	42	3947	jobName.....	53
3901	JmFinishingTC.....	40	3948	jobNaturalLanguageTag.....	52
3902	jmGeneralAttributePersistence.....	78	3949	jobOriginatingHost.....	54
3903	jmGeneralJobPersistence.....	78	3950	jobPriority.....	56
3904	jmGeneralJobSetIndex.....	76	3951	jobProcessAfterDateAndTime.....	56
3905	jmGeneralJobSetName.....	78	3952	jobProcessingCPUTime.....	64
3906	jmGeneralNewestActiveJobIndex.....	77	3953	jobServiceTypes.....	54
3907	jmGeneralNumberOfActiveJobs.....	77	3954	jobSourceChannelIndex.....	54
3908	jmGeneralOldestActiveJobIndex.....	77	3955	jobSourcePlatformType.....	54
3909	jmJobIDJobIndex.....	81	3956	jobStartedBeingHeldTime.....	64
3910	jmJobIDJobSetIndex.....	80	3957	jobStartedProcessingTime.....	64
3911	jmJobImpressionsCompleted.....	85	3958	jobStateReasons2.....	51
3912	jmJobImpressionsPerCopyRequested.....	84	3959	jobStateReasons3.....	51
3913	jmJobIndex.....	82	3960	jobStateReasons4.....	51
3914	jmJobKOctetsPerCopyRequested.....	83	3961	jobSubmissionTime.....	64
3915	jmJobKOctetsProcessed.....	84	3962	jobSubmissionToServerTime.....	64
3916	jmJobOwner.....	85	3963	jobURI.....	52
3917	JmJobServiceTypesTC.....	66	3964	mediumConsumed.....	62
3918	JmJobSourcePlatformTypeTC.....	39	3965	mediumRequested.....	62
3919	jmJobState.....	82	3966	numberOfDocuments.....	55
3920	jmJobStateReasons1.....	83	3967	other.....	50
3921	JmJobStateReasons1TC.....	68	3968	outputBin.....	57
3922	JmJobStateReasons2TC.....	71	3969	pagesCompleted.....	61
3923	JmJobStateReasons3TC.....	74	3970	pagesCompletedCurrentCopy.....	61
3924	JmJobStateReasons4TC.....	75	3971	pagesRequested.....	61
3925	JmJobStateTC.....	47	3972	physicalDevice.....	55

3973	printerResolutionRequested	58	3983	sheetsCompleted	62
3974	printerResolutionUsed	58	3984	sheetsCompletedCurrentCopy	62
3975	printQualityRequested	57	3985	sheetsRequested	61
3976	printQualityUsed	58	3986	sides	57
3977	processingMessage	51	3987	submittingApplicationName	54
3978	processingMessageNaturalLanguageTag	51	3988	submittingServerName	54
3979	queueNameRequested	55	3989	tonerDensityRequested	58
3980	serverAssignedJobName	53	3990	tonerDensityUsed	58
3981	sheetCompletedCopyNumber	59	3991	tonerEcomonyRequested	58
3982	sheetCompletedDocumentNumber	59	3992	tonerEcomonyUsed	58

3993