

Job Monitoring MIB, V0.8887

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

From: Tom Hastings

Date: 12/1103/97

Version: ~~0.88-87~~ (already numbered 1.0 in body, waiting for proof reading)

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

Status: ~~Eleventh and Final Tenth~~ draft MIB that incorporates the agreements reached [at the JMP Meeting, on 12/5/97 in L.A. on the DL](#) on issues in V0.876 which was released after the ~~10/319/49~~ meeting. The changes include:

1. use the new PWG OIDs [without the standard arc](#).
2. make the document a PWG draft standard that will be sent as an Internet-Draft that will become an IETF Informational RFC, [including changing the IANA Considerations section](#) ~~[not done]~~
- ~~3. add natural language support like IPP~~
3. ~~add/fix~~ [the issues with](#) monitoring collated/uncollated implementations ~~[see issues]~~
- ~~4. fix impressions completed;~~
- ~~4. allows multiple Job Submission Id entries to point to the same jmJobIndex entry~~
4. and add ~~3~~[any](#) new Job Submission Ids ~~[not done]~~

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

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

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

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

31 | INTERNET-DRAFT

32

33 |

34

35 |

36

37 |

38

39 |

40

41 |

42 |

43

44

Status of this Memo

45

46

47

48

49

50

51

52

53

54

55

56 |

57

58 |

59

60

61 |

62

63

64

65

66

67

R.~~on~~ Bergman
Dataproducts Corp.T.~~om~~ Hastings
Xerox CorporationS.~~cott~~ Isaacson
Novell, Inc.H.~~arry~~ Lewis
IBM Corp.December 112, 1997

Job Monitoring MIB - V10.87

<draft-ietf-printmib-job-monitor-076.txt>

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.

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."

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).

[This Internet-Draft expires on June 12, 1997.](#)

Abstract

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

68 MIB. Future extensions to this MIB may include, but are not limited to, fax
69 machines and scanners.

70

71

TABLE OF CONTENTS

72	1. INTRODUCTION.....	10
73	1.1 Types of Information in the MIB	10
74	1.2 Types of Job Monitoring Applications	11
75	2. TERMINOLOGY AND JOB MODEL.....	12
76	2.1 System Configurations for the Job Monitoring MIB	15
77	2.1.1 Configuration 1 - client-printer.....	16
78	2.1.2 Configuration 2 - client-server-printer - agent in the server	17
79	2.1.3 Configuration 3 - client-server-printer - client monitors printer agent and server	18
80	3. MANAGED OBJECT USAGE.....	20
81	3.1 Conformance Considerations	20
82	3.1.1 Conformance Terminology.....	20
83	3.1.2 Agent Conformance Requirements.....	20
84	3.1.2.1 MIB II System Group objects	21
85	3.1.2.2 MIB II Interface Group objects	21
86	3.1.2.3 Printer MIB objects	21
87	3.1.3 Job Monitoring Application Conformance Requirements	21
88	3.2 The Job Tables and the Oldest Active and Newest Active Indexes	22
89	3.3 The Attribute Mechanism	23
90	3.3.1 Conformance of Attribute Implementation.....	25
91	3.3.2 Useful, 'Unknown', and 'Other' Values for Objects and Attributes.....	23
92	3.3.3 Data Sub-types and Attribute Naming Conventions.....	26
93	3.3.4 Single-Value (Row) Versus Multi-Value (MULTI-ROW) Attributes	26
94	3.3.5 Requested Attributes	27
95	3.3.6 Consumption Attributes.....	27
96	3.3.7 Index Value Attributes	27
97	3.4 Job Identification	27
98	3.5 Internationalization Considerations	32
99	3.5.1 'JmUTF8StringTC' for text generated by the server or device.....	25
100	3.5.2 'JmJobStringTC' for text generated by the job submitter.....	25
101	3.5.3 'DateAndTime' for representing the date and time.....	25
102	3.6 IANA Considerations	35
103	3.6.1 IANA Registration of enums	35
104	3.6.1.1 Type 1 enumerations.....	35

105	3.6.1.2 Type 2 enumerations.....	35
106	3.6.1.3 Type 3 enumeration	36
107	3.6.2 IANA Registration of type 2 bit values	36
108	3.6.3 IANA Registration of Job Submission Id Formats	36
109	3.6.4 IANA Registration of MIME types/sub-types for document-formats.....	37
110	3.7 Security Considerations	37
111	3.7.1 Read-Write objects	37
112	3.7.2 Read-Only Objects In Other User's Jobs.....	37
113	3.8 Values for Objects	25
114	3.9 Notifications	37
115	4. MIB SPECIFICATION	37
116	Textual conventions for this MIB module	39
117	JmUTF8StringTC	40
118	JmJobStringTC.....	40
119	JmNaturalLanguageTC.....	32
120	JmTimeStampTC.....	40
121	JmJobSourcePlatformTypeTC	40
122	JmFinishingTC	41
123	JmPrintQualityTC.....	42
124	JmPrinterResolutionTC	43
125	JmTonerEconomyTC	43
126	JmBooleanTC.....	43
127	JmMediumTypeTC.....	44
128	JmJobCollationTypeTC.....	37
129	JmJobSubmissionIDTypeTC.....	45
130	JmJobStateTC.....	49
131	JmAttributeTypeTC.....	52
132	other (Int32(-2..) and/or Octets63).....	53
133	Job State attributes.....	53
134	jobStateReasons2 (JmJobStateReasons2TC)	53
135	jobStateReasons3 (JmJobStateReasons3TC)	53
136	jobStateReasons4 (JmJobStateReasons4TC)	53
137	processingMessage (UTF8String63).....	53
138	processingMessageNaturalLanguageTag	53
139	jobCodedCharSet (CodedCharSet)	53
140	jobNaturalLanguageTag	53
141	Job Identification attributes	55
142	jobURI (Octets(1..255)).....	46
143	jobAccountName (Octets63).....	55
144	serverAssignedJobName (JobString63)	55
145	jobName (JobString63).....	56
146	jobServiceTypes (JmJobServiceTypesTC)	56
147	jobSourceChannelIndex (Int32(0..))	57
148	jobSourcePlatformType (JmJobSourcePlatformTypeTC)	57

149	submittingServerName (JobString63).....	57
150	submittingApplicationName (JobString63)	57
151	jobOriginatingHost (JobString63).....	57
152	deviceNameRequested (JobString63)	57
153	queueNameRequested (JobString63)	57
154	physicalDevice (hrDeviceIndex and/or UTF8String63)	58
155	numberOfDocuments (Int32(-2..)).....	58
156	fileName (JobString63).....	58
157	documentName (JobString63)	58
158	jobComment (JobString63).....	58
159	documentFormatIndex (Int32(0..)).....	58
160	documentFormat (PrtInterpreterLangFamilyTC and/or Octets63).....	59
161	Job Parameter attributes.....	59
162	jobPriority (Int32(1..100))	59
163	jobProcessAfterDateAndTime (DateAndTime).....	60
164	jobHold (JmBooleanTC)	60
165	jobHoldUntil (JobString63)	60
166	outputBin (Int32(0..) and/or JobString63)	60
167	sides (Int32(-2..2))	60
168	finishing (JmFinishingTC).....	60
169	Image Quality attributes (requested and used).....	60
170	printQualityRequested (JmPrintQualityTC).....	61
171	printQualityUsed (JmPrintQualityTC).....	61
172	printerResolutionRequested (JmPrinterResolutionTC).....	61
173	printerResolutionUsed (JmPrinterResolutionTC)	61
174	tonerEconomyRequested (JmTonerEconomyTC).....	61
175	tonerEconomyUsed (JmTonerEconomyTC).....	61
176	tonerDensityRequested (Int32(-2..100))	61
177	tonerDensityUsed (Int32(-2..100)).....	61
178	Job Progress attributes (requested and consumed)	61
179	jobCopiesRequested (Int32(-2..)).....	62
180	jobCopiesCompleted (Int32(-2..)).....	62
181	documentCopiesRequested (Int32(-2..))	62
182	documentCopiesCompleted (Int32(-2..))	62
183	jobKOctetsTransferred (Int32(-2..)).....	62
184	sheetCompletedCurrent CopyNumber (Int32(-2..))	51
185	sheetCompletedCurrent DocumentNumber (Int32(-2..))	51
186	jobCollationType (JmCollationTypeTC)	
187	Impression attributes (requested and consumed)	64
188	impressionsSpooled (Int32(-2..))	64
189	impressionsSentToDevice (Int32(-2..)).....	64
190	impressionsInterpreted (Int32(-2..)).....	64
191	impressionsCompletedCurrentCopy (Int32(-2..))	64
192	fullColorImpressionsCompleted (Int32(-2..))	64
193	highlightColorImpressionsCompleted (Int32(-2..)).....	65
194	Page attributes (requested and consumed)	65
195	pagesRequested (Int32(-2..)).....	65
196	pagesCompleted (Int32(-2..)).....	65
197	pagesCompletedCurrentCopy (Int32(-2..))	65
198	Sheet attributes (requested and consumed)	66

199	sheetsRequested (Int32(-2..))	66
200	sheetsCompleted (Int32(-2..))	66
201	sheetsCompletedCurrentCopy (Int32(-2..))	66
202	Resource attributes (requested and consumed)	66
203	mediumRequested (JmMediumTypeTC and/or JobString63)	66
204	mediumConsumed (Int32(-2..) and JobString63)	67
205	colorantRequested (Int32(-2..) and/or JobString63)	67
206	colorantConsumed (Int32(-2..) and/or JobString63)	67
207	Time attributes (set by server or device)	67
208	jobSubmissionToServerTime (JmTimeStampTC and/or DateAndTime)	68
209	jobSubmissionTime (JmTimeStampTC and/or DateAndTime)	68
210	jobStartedBeingHeldTime (JmTimeStampTC and/or DateAndTime)	68
211	jobStartedProcessingTime (JmTimeStampTC and/or DateAndTime)	69
212	jobCompletionTime (JmTimeStampTC and/or DateAndTime)	69
213	jobProcessingCPUTime (Int32(-2..))	69
214	JmJobServiceTypesTC	71
215	JmJobStateReasons1TC	72
216	JmJobStateReasons2TC	76
217	JmJobStateReasons3TC	79
218	JmJobStateReasons4TC	79
219	The General Group (MANDATORY)	81
220	jmGeneralJobSetIndex (Int32(1..32767))	81
221	jmGeneralNumberOfActiveJobs (Int32(0..))	82
222	jmGeneralOldestActiveJobIndex (Int32(0..))	82
223	jmGeneralNewestActiveJobIndex (Int32(0..))	82
224	jmGeneralJobPersistence (Int32(15..))	83
225	jmGeneralAttributePersistence (Int32(15..))	83
226	jmGeneralJobSetName (UTF8String63)	84
227	The Job ID Group (MANDATORY)	84
228	jmJobSubmissionID (OCTET STRING(SIZE(48)))	85
229	jmJobIDJobSetIndex (Int32(1..32767))	86
230	jmJobIDJobIndex (Int32(1..))	86
231	The Job Group (MANDATORY)	86
232	jmJobIndex (Int32(1..))	87
233	jmJobState (JmJobStateTC)	88
234	jmJobStateReasons1 (JmJobStateReasons1TC)	88
235	jmNumberOfInterveningJobs (Int32(-2..))	88
236	jmJobKOctetsPerCopyRequested (Int32(-2..))	89
237	jmJobKOctetsProcessed (Int32(-2..))	89
238	jmJobImpressionsPerCopyRequested (Int32(-2..))	90
239	jmJobImpressionsCompleted (Int32(-2..))	90
240	jmJobOwner (JobString63)	90
241	The Attribute Group (MANDATORY)	91
242	jmAttributeTypeIndex (JmAttributeTypeTC)	92
243	jmAttributeInstanceIndex (Int32(1..32767))	93
244	jmAttributeValueAsInteger (Int32(-2..))	93

245 jmAttributeValueAsOctets (Octets63)..... 94

246 **5. APPENDIX A - IMPLEMENTING THE JOB LIFE CYCLE..... 97**

247 **6. APPENDIX B - SUPPORT OF THE JOB SUBMISSION ID IN JOB**
248 **SUBMISSION PROTOCOLS..... 97**

249 6.1 Hewlett-Packard's Printer Job Language (PJL) 98

250 6.2 ISO DPA 98

251 **7. REFERENCES 98**

252 **8. AUTHOR'S ADDRESSES 100**

253 **9. INDEX..... 103**
254

255

Job Monitoring MIB

256 1. Introduction

257 This specification was developed and approved by the Printer Working Group (PWG) as
258 a PWG standard for an SNMP MIB. See <http://www.pwg.org>. In consultation with the
259 IETF Application Area Directors, it was concluded that this MIB should not be entered
260 on the Internet standards track, because this MIB does not facilitate the management of
261 the network itself. This MIB is limited to the management of networked printers.
262 Therefore, the SNMP OBJECT IDENTIFIERS have been assigned under the enterprises
263 arc, using the number assignment given to the PWG organization.

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

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

284 1.1 Types of Information in the MIB

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

287 User:

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

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

295 Operator:

296 Provide a presentation of the state of all the jobs in the print system.
297 Provide the ability to identify the user that submitted the print job.
298 Provide the ability to identify the resources required by each job.
299 Provide the ability to define which physical printers are candidates for the print
300 job.
301 Provide some idea of how long each job will take. However, exact estimates of
302 time to process a job is not being attempted. Instead, objects are included that
303 allow the operator to be able to make gross estimates.

304 Capacity Planner:

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

307 Accountant:

308 Provide information to allow the creation of a record of resources consumed and
309 printer usage data for charging users or groups for resources consumed.
310 Provide information to allow the prediction of consumable usage and resource
311 need.

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

317 1.2 Types of Job Monitoring Applications

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

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

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

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

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

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

353 2. Terminology and Job Model

354 This section defines the terms that are used in this specification and the general model for
355 jobs [in alphabetical order](#).

356 NOTE - Existing systems use conflicting terms, so these terms are drawn from the ISO
357 10175 Document Printing Application (DPA) standard[iso-dpa]. For example,
358 PostScript systems use the term *session* for what is called a *job* in this specification
359 and the term *job* to mean what is called a *document* in this specification.

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

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

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

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

379 Collated Documents: A job collation type in which each copy of a job contains a single
380 copy of each document and in the order of the document(s) in the job. The sheets within
381 each document copy are also collated internally within the device (so called "mopier").
382 The document copies within the job are collated by making multiple passes over all the
383 document(s) in the job as a whole, either the original representation or an intermediate
384 form. For example, if a job is submitted with documents, A and B, the job is produced as
385 A, B, A, B, This job collation type corresponds to the IPP [ipp-model] 'separate-
386 documents-collated-copies' value of the "multiple-document-handling" attribute. See
387 "job collation" and "uncollated documents".

388 Collated Sheets: Each sheet in a document copy occurs in the order of the document and
389 occurs only once in each document copy. It is not an enumerated Job Collation Type, but
390 is the opposite of the Uncollated Sheets job collation type. See the definitions of the
391 "collated documents" and "uncollated documents" job collation types, which both have
392 collated sheets. See also "uncollated sheets".

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

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

399 Document Instruction: An instruction specifying how to process the document.

400 Document instructions MAY be passed in the job submission protocol separate from the

401 actual document data, or MAY be embedded in the document data or a combination,
402 depending on the job submission protocol and implementation.

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

404 Impression: For a print job, an impression is the passage of the entire side of a sheet by
405 the marker, whether or not any marks are made and independent of the number of passes
406 that the side makes past the marker. Thus a four pass color process counts as a single
407 impression. One-sided processing involves one impression per sheet. Two-sided
408 processing involves two impressions per sheet. If a two-sided document has an odd
409 number of pages, the last sheet still counts as two impressions, if that sheet makes two
410 passes through the marker or the marker marks on both sides of a sheet in a single pass.
411 Two-up printing is the placement of two logical pages on one side of a sheet and so is still
412 a single impression. See "page" and "sheet".

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

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

417 Job Collation: The specification of the order of sheets within document copies and
418 documents copies within job copies. See "collated documents", "uncollated documents"
419 and "uncollated sheets", which are the three types of Job Collation.

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

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

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

431 Job Set: A group of jobs that are queued and scheduled together according to a specified
432 scheduling algorithm for a specified device or set of devices. For implementations that
433 embed the SNMP agent in the device, the MIB job set normally represents *all* the jobs
434 known to the device, so that the implementation only implements a single job set. If the
435 SNMP agent is implemented in a server that controls one or more devices, each MIB job
436 set represents a job queue for (1) a specific device or (2) set of devices, if the server uses

437 a single queue to load balance between several devices. Each job set is disjoint; no job
438 SHALL be represented in more than one MIB job set.

439 Monitor: Short for Job Monitoring Application.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

472 Uncollated Documents: A job collation type in which each copy of a document that
473 contains multiple documents are grouped together and in the order that the documents
474 occur in the job. The sheets within each document copy are also collated internally
475 within the device (so called "mopier") by making multiple passes over each document in
476 the job separately, either the original representation or an intermediate form. For
477 example, if a job is submitted with documents, A and B, the job is produced as A, A, ...,
478 B, B, This job collation type corresponds to the IPP [ipp-model] 'separate-
479 documents-uncollated-copies' value of the "multiple-document-handling" attribute. If the
480 job has only one document or only one copy of multiple documents, there is no
481 distinction between 'Collated Documents' and "Uncollated Documents', so the latter
482 SHALL NOT be designated. See "job collation" and "collated documents".

483 Uncollated Sheets: A job collation type in which each sheet of a document that is to
484 produce multiple copies is replicated before the next sheet in the document is processed
485 and stacked. If the device has an output bin collator, uncollated sheets may actually
486 produce collated sheets as far as the user is concerned (in the output bins). However,
487 when the job collation is 'uncollated sheets', job progress is indistinguishable to a
488 monitoring application between a device that has an output bin collator and one that does
489 not. See "job collation".

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

491 **2.1 System Configurations for the Job Monitoring MIB**

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

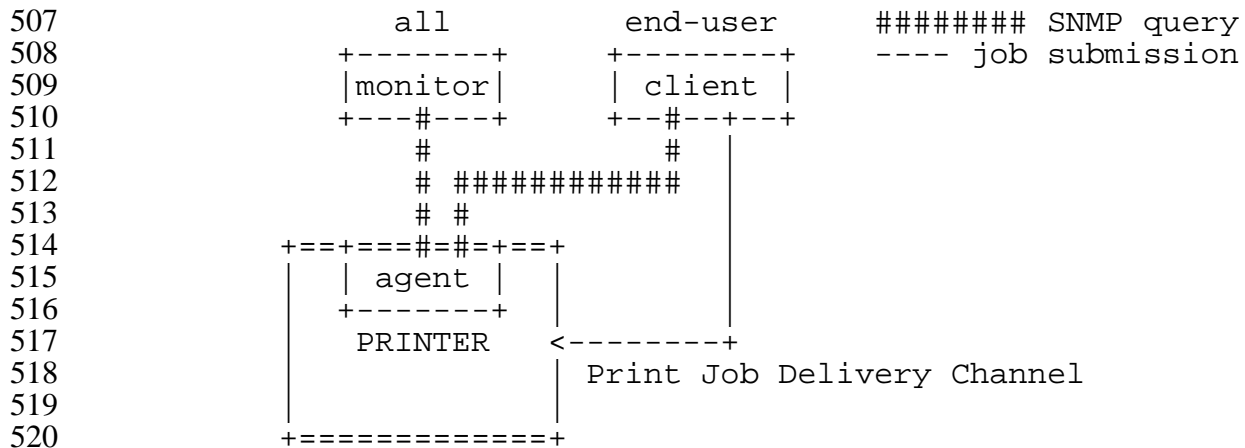
495 The diagram in the Printer MIB[print-mib] entitled: "One Printer's View of the Network"
496 is assumed for this MIB as well. Please refer to that diagram to aid in understanding the
497 following system configurations.

498 **2.1.1 Configuration 1 - client-printer**

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

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

506



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

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

- 524 1. Multiple **clients** MAY submit jobs to a **printer**.
- 525 2. Multiple **clients** MAY monitor a **printer**.
- 526 3. Multiple **monitors** MAY monitor a **printer**.
- 527 4. A **client** MAY submit jobs to multiple **printers**.
- 528 5. A **monitor** MAY monitor multiple **printers**.

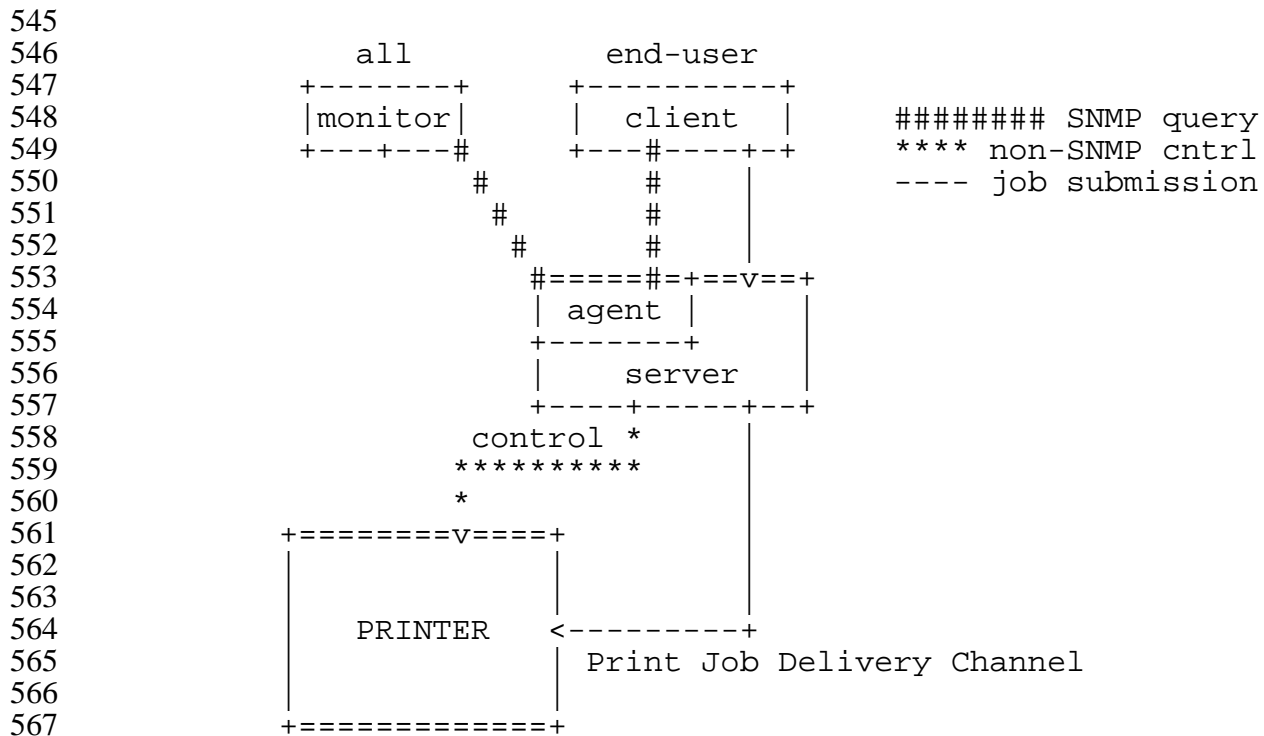
529 2.1.2 Configuration 2 - client-server-printer - agent in the server

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

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

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

536 There is no SNMP Job Monitoring MIB agent in the **printer** in configuration 2, at least
537 that the client or monitor are aware. In this configuration, the agent SHALL return the
538 current values of the objects in the Job Monitoring MIB both for jobs the server keeps
539 and jobs that the server has submitted to the **printer**. The Job Monitoring MIB agent
540 SHALL obtain the required information from the **printer** by a method that is beyond the
541 scope of this document. The agent in the **server** SHALL keep the job in the Job
542 Monitoring MIB in the server as long as the job is in the **printer**, plus a defined time
543 period after the job enters the **completed** state in which accounting programs can copy
544 out the accounting data from the Job Monitoring MIB.



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

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

- 571 1. Multiple **clients** MAY submit jobs to a **server**.
- 572 2. Multiple **clients** MAY monitor a **server**.
- 573 3. Multiple **monitors** MAY monitor a **server**.
- 574 4. A **client** MAY submit jobs to multiple **servers**.
- 575 5. A **monitor** MAY monitor multiple **servers**.
- 576 6. Multiple **servers** MAY submit jobs to a **printer**.
- 577 7. Multiple **servers** MAY control a **printer**.

578 2.1.3 Configuration 3 - client-server-printer - client monitors printer agent and 579 server

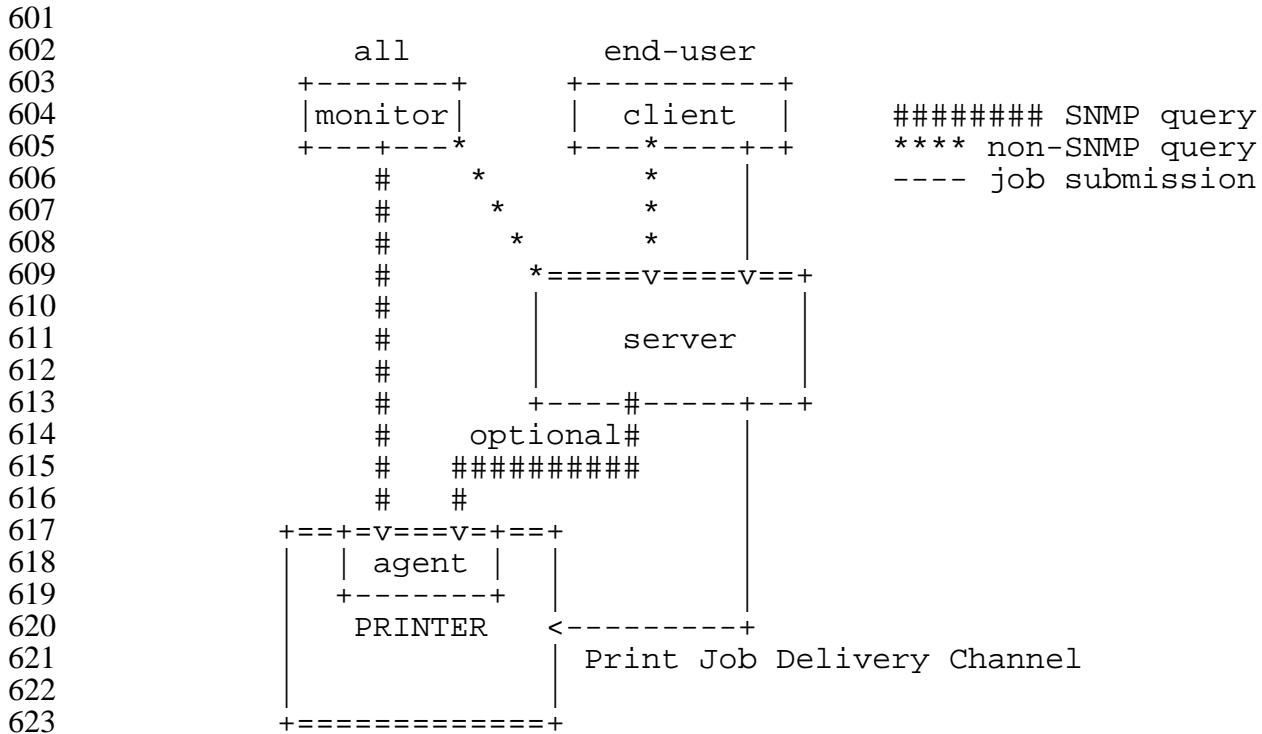
580 In the **client-server-printer** configuration 3, the **client(s)** submit jobs to an intermediate
 581 **server** by some network connection, *not* directly to the **printer**. That server does *not*
 582 contain a Job Monitoring MIB agent.

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

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

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

592 In configuration 3, the agent (in the **printer**) SHALL keep the values of the objects in the
 593 Job Monitoring MIB that the agent implements updated for a job that the server has
 594 submitted to the printer. The agent SHALL obtain information about the jobs submitted
 595 to the printer from the server (either in the job submission protocol, in the document data,
 596 or by direct query of the server), in order to populate some of the objects the Job
 597 Monitoring MIB in the printer. The agent in the printer SHALL keep the job in the Job
 598 Monitoring MIB as long as the job is in the Printer, and longer in order to implement the
 599 **completed** state in which monitoring programs can copy out the accounting data from the
 600 Job Monitoring MIB.



624 **Figure 2-3 - Configuration 3 - client-server-printer - client monitors printer agent**
 625 **and server**

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

- 628 1. Multiple **clients** MAY submit jobs to a **server**.
 629 2. Multiple **clients** MAY monitor a **server**.
 630 3. Multiple **monitors** MAY monitor a **server**.

- 631 4. A **client** MAY submit jobs to multiple **servers**.
632 5. A **monitor** MAY monitor multiple **servers**.
633 6. Multiple **servers** MAY submit jobs to a **printer**.
634 7. Multiple **servers** MAY control a **printer**.

635 3. Managed Object Usage

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

637 3.1 Conformance Considerations

638 In order to achieve interoperability between job monitoring applications and job
639 monitoring agents, this specification includes the conformance requirements for both
640 monitoring applications and agents.

641 3.1.1 Conformance Terminology

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

- 644 • "SHALL": indicates an action that the subject of the sentence must implement in
645 order to claim conformance to this specification
- 646 • "MAY": indicates an action that the subject of the sentence does not have to
647 implement in order to claim conformance to this specification, in other words that
648 action is an implementation option
- 649 • "NEED NOT": indicates an action that the subject of the sentence does not have to
650 implement in order to claim conformance to this specification. The verb "NEED
651 NOT" is used instead of "may not", since "may not" sounds like a prohibition.
- 652 • "SHOULD": indicates an action that is recommended for the subject of the
653 sentence to implement, but is not required, in order to claim conformance to this
654 specification.

655 3.1.2 Agent Conformance Requirements

656 A conforming agent:

- 657 1. SHALL implement *all* MANDATORY groups in this specification.
- 658 2. SHALL implement any attributes if (1) the server or device supports the
659 functionality represented by the attribute and (2) the information is available
660 to the agent.
- 661 3. SHOULD implement both forms of an attribute if it implements an attribute
662 that permits a choice of INTEGER and OCTET STRING forms, since
663 implementing both forms may help management applications by giving them

664 a choice of representations, since the representation are equivalent. See the
665 **JmAttributeTypeTC** textual-convention.

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

668 3.1.2.1 MIB II System Group objects

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

671 3.1.2.2 MIB II Interface Group objects

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

674 3.1.2.3 Printer MIB objects

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

681 3.1.3 Job Monitoring Application Conformance Requirements

682 A conforming job monitoring application:

- 683 1. SHALL accept the full syntactic range for all objects in all MANDATORY
684 groups and all MANDATORY attributes that are required to be implemented
685 by an agent according to Section 3.1.2 and SHALL either present them to the
686 user or ignore them.
- 687 2. SHALL accept the full syntactic range for *all* attributes, including enum and
688 bit values specified in this specification and additional ones that may be
689 registered with IANA and SHALL either present them to the user or ignore
690 them. In particular, a conforming job monitoring application SHALL not
691 malfunction when receiving any standard or registered enum or bit values.
692 See Section 3.7 entitled "IANA [and PWG Registration](#) Considerations".
- 693 3. SHALL NOT fail when operating with agents that materialize attributes *after*
694 the job has been submitted, as opposed to when the job is submitted.
- 695 4. SHALL, if it supports a time attribute, accept either form of the time attribute,
696 since agents are free to implement either time form.

697 **3.2 The Job Tables and the Oldest Active and Newest Active Indexes**

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

- 700 1. **jmGeneralJobSetIndex** - which job set
701 2. **jmJobIndex** - which job in the job set

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

- 704 1. **jmGeneralOldestActiveJobIndex** - the index of the active job that has been
705 in the tables the longest.
706 2. **jmGeneralNewestActiveJobIndex** - the index of the active job that has been
707 most recently added to the tables.

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

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

718 It is recommended that agents that are providing access to servers/devices that already
719 allocate job-identifiers for jobs as integers use the same integer value for the
720 **jmJobIndex**. Then management applications using this MIB and applications using
721 other protocols will see the same job identifiers for the same jobs. Agents providing
722 access to systems that contain jobs with a job identifier of **0** SHALL map the job
723 identifier value **0** to a **jmJobIndex** value that is one higher than the highest job identifier
724 value that any job can have on that system. Then only job 0 will have a different job-
725 identifier value than the job's **jmJobIndex** value.

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

731 Each time a new job is accepted by the server or device that the agent is providing access
732 to AND that job is to be 'active' (**pending**, **processing**, or **processingStopped**, but not
733 **pendingHeld**), the agent SHALL copy the value of the job's **jmJobIndex** to the
734 **jmGeneralNewestActiveJobIndex** object. If the new job is to be 'inactive'

735 (**pendingHeld** state), the agent SHALL not change the value of
736 **jmGeneralNewestActiveJobIndex** object (though the agent SHALL assign the next
737 incremental **jmJobIndex** value to the job).

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

744 Whenever a job transitions from one of the 'inactive' job states to one of the 'active' job
745 states (from **pendingHeld** to **pending** or **processing**), the agent SHALL update the value
746 of either the **jmGeneralOldestActiveJobIndex** or the
747 **jmGeneralNewestActiveJobIndex** objects, or both, if the job's **jmJobIndex** value is
748 outside the range between **jmGeneralOldestActiveJobIndex** and
749 **jmGeneralNewestActiveJobIndex**.

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

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

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

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

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

768 3.3 The Attribute Mechanism

769 Attributes are similar to information objects, except that attributes are identified by an
770 enum, instead of an OID, so that attributes may be registered without requiring a new

771 MIB. Also an implementation that does not have the functionality represented by the
772 attribute can omit the attribute entirely, rather than having to return a distinguished value.
773 The agent is free to materialize an attribute in the **jmAttributeTable** as soon as the agent
774 is aware of the value of the attribute.

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

- 776 1. **jmGeneralJobSetIndex** - which job set
- 777 2. **jmJobIndex** - which job in the job set
- 778 3. **jmAttributeTypeIndex** - which attribute
- 779 4. **jmAttributeInstanceIndex** - which attribute instance for those attributes that
780 can have multiple values per job.

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

788 NOTE - The table of contents lists all the attributes in order. This order is the order of
789 enum assignments which is the order that the SNMP GetNext operation returns attributes.
790 Most attributes apply to all three configurations covered by this MIB specification (see
791 section 0 entitled "[Two-up: The placement of two pages on one side of a sheet so that](#)
792 [each side or impressions counts as two pages. See "page" and "sheet".](#)

793 [Uncollated Documents: A job collation type in which each copy of a document that](#)
794 [contains multiple documents are grouped together and in the order that the documents](#)
795 [occur in the job. The sheets within each document copy are also collated internally](#)
796 [within the device \(so called "mopier"\) by making multiple passes over each document in](#)
797 [the job separately, either the original representation or an intermediate form. For](#)
798 [example, if a job is submitted with documents, A and B, the job is produced as A, A, ...,](#)
799 [B, B, This job collation type corresponds to the IPP \[ipp-model\] 'separate-](#)
800 [documents-uncollated-copies' value of the "multiple-document-handling" attribute. If the](#)
801 [job has only one document or only one copy of multiple documents, there is no](#)
802 [distinction between 'Collated Documents' and 'Uncollated Documents', so the latter](#)
803 [SHALL NOT be designated. See "job collation" and "collated documents".](#)

804 [Uncollated Sheets: A job collation type in which each sheet of a document that is to](#)
805 [produce multiple copies is replicated before the next sheet in the document is processed](#)
806 [and stacked. If the device has an output bin collator, uncollated sheets may actually](#)
807 [produce collated sheets as far as the user is concerned \(in the output bins\). However,](#)
808 [when the job collation is 'uncollated sheets', job progress is indistinguishable to a](#)

809 [monitoring application between a device that has an output bin collator and one that does](#)
810 [not. See "job collation".](#)

811 User: A person that uses a client or a monitor. [See "end user".](#)

812 System Configurations for the Job Monitoring MIB"). Those attributes that apply to a
813 particular configuration are indicated as '**Configuration n:**' and SHALL NOT be used
814 with other configurations.

815 3.3.1 Conformance of Attribute Implementation

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

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

824 3.3.2 Useful, 'Unknown', and 'Other' Values for Objects and Attributes

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

829 SNMP requires that if an object cannot be implemented because its values cannot be
830 accessed, then a compliant agent SHALL return an SNMP error in SNMPv1 or an
831 exception value in SNMPv2. However, this MIB has been designed so that 'all' objects
832 can and SHALL be implemented by an agent, so that neither the SNMPv1 error nor the
833 SNMPv2 exception value SHALL be generated by the agent. This MIB has also been
834 designed so that when an agent materializes an attribute, the agent SHALL materialize a
835 row consisting of both the **jmAttributeValueAsInteger** and **jmAttributeValueAsOctets**
836 objects.

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

842 Since each attribute is represented by a row consisting of both the
843 **jmAttributeValueAsInteger** and **jmAttributeValueAsOctets** MANDATORY objects,

844 SNMP requires that the agent SHALL always create an attribute row with both objects
 845 specified. However, for most attributes the agent SHALL return a "useful" value for one
 846 of the objects and SHALL return the 'other' value for the other object. For integer only
 847 attributes, the agent SHALL always return a zero-length string value for the
 848 **jmAttributeValueAsOctets** object. For octet string only attributes, the agent SHALL
 849 always return a '-1' value for the **jmAttributeValueAsInteger** object.

850 3.3.3 Data Sub-types and Attribute Naming Conventions

851 Many attributes are sub-typed to give a more specific data type than **Integer32** or
 852 **OCTET STRING**. The data sub-type of each attribute is indicated on the first line(s) of
 853 the description. Some attributes have several different data sub-type representations.
 854 When an attribute has both an **Integer32** data sub-type and an **OCTET STRING** data
 855 sub-type, the attribute can be represented in a single row in the **jmAttributeTable**. In
 856 this case, the data sub-type name is not included as the last part of the name of the
 857 attribute, e.g., **documentFormat(38)** which is both an enum and/or a name. When the
 858 data sub-types cannot be represented by a single row in the **jmAttributeTable**, each such
 859 representation is considered a separate attribute and is assigned a separate name and enum
 860 value. For these attributes, the name of the data sub-type is the last part of the name of
 861 the attribute: **Name**, **Index**, **DateAndTime**, **TimeStamp**, etc. For example,
 862 **documentFormatIndex(37)** is an index.

863 NOTE: The Table of Contents also lists the data sub-type and/or data sub-types of each
 864 attribute, using the textual-convention name when such is defined. The following
 865 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.

866 3.3.4 Single-Value (Row) Versus Multi-Value (MULTI-ROW) Attributes

867 Most attributes SHALL have only one row per job. However, a few attributes can have
 868 multiple values per job or even per document, where each value is a separate row in the
 869 **jmAttributeTable**. Unless indicated with **'MULTI-ROW:'** in the **JmAttributeTypeTC**
 870 description, an agent SHALL ensure that each attribute occurs only once in the
 871 **jmAttributeTable** for a job. Most of the **'MULTI-ROW'** attributes do not allow
 872 duplicate values, i.e., the agent SHALL ensure that each value occurs only once for a job.

873 Only if the specification of the **MULTI-ROW** attribute also says "the values NEED
874 NOT be unique" can the agent allow duplicate values to occur for the job.

875 NOTE - Duplicates are allowed for 'extensive' **MULTI-ROW** attributes, such as
876 **fileName(34)** or **documentName(35)** which are specified to be 'per-document' attributes,
877 but are *not* allowed for 'intensive' **MULTI-ROW** attributes, such as
878 **mediumConsumed(171)** and **documentFormat(38)** which are specified to be 'per-job'
879 attributes.

880 3.3.5 Requested Objects and Attributes

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

889 3.3.6 Consumption Attributes

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

896 3.3.7 Index Value Attributes

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

903 3.4 Monitoring Job Progress

904 There are a number of objects and attributes for monitoring the progress of a job. These
905 objects and attributes count the number of K octets, impressions, sheets, and pages
906 requested or completed, i.e., processed or stacked, depending on implementation. For

907 impressions and sheets, "completed" SHALL mean stacked, unless the implementation is
 908 unable to detect when each sheet is stacked, in which case stacked is approximated when
 909 processing of each sheet completes. There are objects and attributes for the overall job
 910 and for the current copy of the document currently being ~~processed or~~ stacked. For the
 911 latter, the rate at which the various objects and attributes count depends on the sheet and
 912 document collation of the job.

913 Job Collation included sheet collation and document collation. Sheet collation is defined
 914 to be the orderingcollations of sheets within a document copy. Document collation is
 915 defined to be orderingcollation of document copies within a multi-document job. There
 916 are three ~~combinations of these two~~ types of job collation (see terminology definitions in
 917 Section 2):

- 918 1. Uncollated Sheets~~External Sheet Collation~~
- 919 2. ~~Internal Sheet Collation with~~ Collated Documents
- 920 3. ~~Internal Sheet Collation with~~ Uncollated Documents

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

- 923 1. **jmJobImpressionsCompleted** - counts the total number of impressions
 924 stacked for the job
- 925 2. **impressionsCompletedCurrentCopy** - counts the number of impressions
 926 stacked for the current document copy
- 927 3. **sheetCompletedcurrentCopyNumber** - identifies the number of the copy for
 928 the current document being stacked where the first copy is 1.
- 929 4. **sheetCompletedcurrentDocumentNumber** - identifies the current document
 930 within the job that is being stacked where the first document in a job is 1.
 931 NOTE: this attribute SHOULD NOT be implemented for implementations
 932 that only support one document per job.

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

936 Job Ceollation Ttype = Uncollated Sheets~~External Sheet Collation~~

937

jmJobImpressions Completed	impressionsCompleted CurrentCopy	sheetCompleted currentCopyNu mber	sheetCompleteddeu rrent DocumentNumber
<u>0</u> 1	<u>0</u> 1	<u>0</u> 1	<u>0</u> 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

938

939 | Job Collation Type = Collated Documents~~Internal Collation with document collated~~
 940 | ~~within each job copy~~

941

<u>jmJobImpressions Completed</u>	<u>impressionsCompleted CurrentCopy</u>	<u>sheetCompleted currentCopyNumber</u>	<u>sheetCompleted current DocumentNumber</u>
<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>
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

942

943 Job Collation Type = Internal Collation with Uncollated Document-copies

944

<u>jmJobImpressions Completed</u>	<u>impressionsCompleted CurrentCopy</u>	<u>sheetCompleted currentCopyNumber</u>	<u>sheetCompleted current DocumentNumber</u>
<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>
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

945

946 For two-sided printing, impressions are defined to include the number of sides that pass
 947 by the marker whether marked or not (see the definition of "impression" in Section 2).
 948 Therefore, documents with an odd number of pages will count an extra impression and
 949 will appear the same as a document with one more page. Also the impression counts will
 950 count by twos and the odd rows in the above tables do not appear below:

951 Job Collation Type = Uncollated Sheets

952

<u>jmJobImpressions Completed</u>	<u>impressionsCompleted CurrentCopy</u>	<u>sheetCompleted CopyNumber</u>	<u>sheetCompleted DocumentNumber</u>
<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>
<u>6</u>	<u>2</u>	<u>2</u>	<u>1</u>
<u>8</u>	<u>2</u>	<u>3</u>	<u>1</u>
<u>10</u>	<u>4</u>	<u>2</u>	<u>1</u>
<u>12</u>	<u>4</u>	<u>3</u>	<u>1</u>
<u>18</u>	<u>2</u>	<u>2</u>	<u>2</u>
<u>20</u>	<u>2</u>	<u>3</u>	<u>2</u>
<u>22</u>	<u>4</u>	<u>2</u>	<u>2</u>

953

954 Job Collation Type = Collated Documents

955

<u>jmJobImpressions Completed</u>	<u>impressionsCompleted CurrentCopy</u>	<u>sheetCompleted CopyNumber</u>	<u>sheetCompleted DocumentNumber</u>
<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>
<u>2</u>	<u>2</u>	<u>1</u>	<u>1</u>
<u>4</u>	<u>4</u>	<u>1</u>	<u>1</u>
<u>6</u>	<u>2</u>	<u>1</u>	<u>2</u>
<u>8</u>	<u>1</u>	<u>2</u>	<u>1</u>
<u>10</u>	<u>2</u>	<u>2</u>	<u>1</u>
<u>12</u>	<u>4</u>	<u>2</u>	<u>1</u>
<u>14</u>	<u>2</u>	<u>2</u>	<u>2</u>
<u>16</u>	<u>4</u>	<u>2</u>	<u>2</u>
<u>18</u>	<u>2</u>	<u>3</u>	<u>1</u>
<u>20</u>	<u>4</u>	<u>3</u>	<u>1</u>
<u>22</u>	<u>2</u>	<u>3</u>	<u>2</u>
<u>24</u>	<u>4</u>	<u>3</u>	<u>2</u>

956

957 Job Collation Type = Uncollated Documents

958

<u>jmJobImpressions Completed</u>	<u>impressionsCompleted CurrentCopy</u>	<u>sheetCompleted CopyNumber</u>	<u>sheetCompleted DocumentNumber</u>
<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>
<u>2</u>	<u>2</u>	<u>1</u>	<u>1</u>
<u>4</u>	<u>4</u>	<u>1</u>	<u>1</u>
<u>6</u>	<u>2</u>	<u>2</u>	<u>1</u>
<u>8</u>	<u>4</u>	<u>2</u>	<u>1</u>
<u>10</u>	<u>2</u>	<u>3</u>	<u>1</u>
<u>12</u>	<u>4</u>	<u>3</u>	<u>1</u>
<u>14</u>	<u>2</u>	<u>1</u>	<u>2</u>
<u>16</u>	<u>4</u>	<u>1</u>	<u>2</u>
<u>18</u>	<u>2</u>	<u>2</u>	<u>2</u>
<u>20</u>	<u>4</u>	<u>2</u>	<u>2</u>
<u>22</u>	<u>2</u>	<u>3</u>	<u>2</u>
<u>24</u>	<u>4</u>	<u>3</u>	<u>2</u>

959

960 3.5 Job Identification

961 There are a number of attributes that permit a user, operator or system administrator to
962 identify jobs of interest, such as **jobURI**, **jobName**, **jobOriginatingHost**, etc. In
963 addition, there is a **jmJobSubmissionID** object that is a text string table index. Being a
964 table index allows a monitoring application to quickly locate and identify a particular job
965 of interest that was submitted from a particular client by the user invoking the monitoring
966 application without having to scan the entire job table. The Job Monitoring MIB needs to
967 provide for identification of the job at both sides of the job submission process. The
968 primary identification point is the client side. The **jmJobSubmissionID** allows the
969 monitoring application to identify the job of interest from all the jobs currently "known"
970 by the server or device. The value of **jmJobSubmissionID** can be assigned by either the
971 client's local system or a downstream server or device. The point of assignment depends
972 on the job submission protocol in use.

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

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

983 In some configurations there may be more than one application program that monitors the
984 same job when the job passes from one network entity to another when it is submitted.
985 See configuration 3. ~~When there are multiple job submission IDs in such a case, each~~
986 ~~entity MAY supply an appropriate~~ **application can have its own jmJobSubmissionID**
987 value. In this case there would be a separate entry in the **jmJobSubmissionID** table, one
988 for each **jmJobSubmissionID**. ~~AllBoth~~ entries would map to the same **jmJobIndex** that
989 contains the job data. When the job is deleted, it is up to the agent to remove ~~allboth~~
990 entries ~~that point to the job~~ from the **jmJobSubmissionID** table as well.

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

994 3.6 Internationalization Considerations

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

996 3.6.1 Text generated by the server or device

997 There are a few objects and attributes generated by the server or device that SHALL be
998 represented using the Universal Multiple-Octet Coded Character Set (UCS) [ISO-10646].
999 These objects and attributes are always supplied (if implemented) by the agent, not by the
1000 job submitting client:

- 1001 1. jmGeneralJobSetName object
- 1002 2. processingMessage(6) attribute
- 1003 3. physicalDevice(32) (name value) attribute

1004 The character encoding scheme for representing these objects and attributes SHALL be
1005 UTF-8 as recommended by RFC 2130 [RFC 2130] and the "IETF Policy on Character
1006 Sets and Language" [char-set policy]. The 'JmUTF8StringTC' textual convention is used
1007 to indicate UTF-8 text strings.

1008 NOTE - For strings in 7-bit US-ASCII, there is no impact since the UTF-8 representation
1009 of 7-bit ASCII is identical to the US-ASCII [US-ASCII] encoding.

1010 The text contained in the **processingMessage(6)** attribute is generated by the
1011 server/device. The natural language for the **processingMessage(6)** attribute is identified
1012 by the **processingMessageNaturalLanguageTag(7)** attribute. The
1013 **processingMessageNaturalLanguageTag(7)** attribute uses the
1014 **JmNaturalLanguageTagTC** textual convention which SHALL conform to the language
1015 tag mechanism specified in RFC 1766 [RFC-1766]. The **JmNaturalLanguageTagTC**
1016 value is the same as the IPP [IPP-model] **naturalLanguage**' attribute syntax. RFC 1766
1017 specifies that a US-ASCII string consisting of the natural language followed by an
1018 optional country field. Both fields use the same two-character codes from ISO 639 [ISO-
1019 639] and ISO 3166 [ISO-3166], respectively, that are used in the Printer MIB for
1020 identifying language and country.

1021 Examples of the values of the **processingMessageNaturalLanguageTag(7)** attribute
1022 include:

- 1023 1. 'en' for English
- 1024 2. 'en-us' for US English
- 1025 3. 'fr' for French
- 1026 4. 'de' for German

1027 3.6.2 Text supplied by the job submitter

1028 All of the objects and attributes represented by the **JmJobStringTC**' textual-convention
1029 are either (1) supplied in the job submission protocol by the client that submits the job to
1030 the server or device or (2) are defaulted by the server or device if the job submitting client
1031 does not supply values. The agent SHALL represent these objects and attributes in the
1032 MIB either (1) in the coded character set as they were submitted or (2) MAY convert the
1033 coded character set to another coded character set or encoding scheme. In any case, the

1034 resulting coded character set representation SHOULD be UTF-8 [UTF-8], but SHALL be
1035 one in which the code positions from 0 to 31 SHALL not be used, 32 to 127 SHALL be
1036 US-ASCII [US-ASCII], 127 SHALL be unused, and the remaining code positions 128 to
1037 255 SHALL represent single-byte or multi-byte graphic characters structured according to
1038 ISO 2022 [ISO 2022] or SHALL be unused.

1039 The coded character set SHALL be one of the ones registered with IANA [IANA] and
1040 SHALL be identified by the **jobCodedCharSet** attribute in the **jmJobAttributeTable** for
1041 the job. If the agent does not know what coded character set was used by the job
1042 submitting client, the agent SHALL either (1) return the **unknown(2)**' value for the
1043 **jobCodedCharSet** attribute or (2) not return the **jobCodedCharSet** attribute for the job.

1044 Examples of coded character sets which meet this criteria for use as the value of the
1045 **jobCodedCharSet** job attribute are: US-ASCII [US-ASCII], ISO 8859-1 (Latin-1) [ISO
1046 8859-1], any ISO 8859-n, HP Roman8, IBM Code Page 850, Windows Default 8-bit set,
1047 UTF-8 [UTF-8], US-ASCII plus JIS X0208-1990 Japanese [JIS X0208], US-ASCII plus
1048 GB2312-1980 PRC Chinese [GB2312]. See the IANA registry of coded character sets
1049 [IANA charsets].

1050 Examples of coded character sets which do not meet this criteria are: national 7-bit sets
1051 conforming to ISO 646 (except US-ASCII), EBCDIC, and ISO 10646 (Unicode) [ISO-
1052 10646]. In order to represent Unicode characters, the UTF-8 [UTF-8] encoding scheme
1053 SHALL be used which has been assigned the MIBenum value of '106' by IANA.

1054 The **jobCodedCharSet** attribute uses the imported '**CodedCharSet**' textual-convention
1055 from the Printer MIB [printmib].

1056 The natural language for all attributes represented by the textual-convention
1057 **JmJobStringTC** SHALL be identified by the **jobNaturalLanguageTag(8)** attribute.
1058 The **jobNaturalLanguageTag(8)** attribute value SHALL have the same syntax and
1059 semantics as the **processingMessageNaturalLanguageTag(7)** attribute, except that the
1060 **jobNaturalLanguageTag(8)** attribute identifies the natural language of attributes
1061 supplied by the job submitter instead of the natural language of the
1062 **processingMessage(6)** attribute. See Section 3.6.13.5.1.

1063 3.6.3 'DateAndTime' for representing the date and time

1064 This MIB also contains objects that are represented using the **DateAndTime** textual
1065 convention from SMIV2 [SMIV2-TC]. The job management application SHALL display
1066 such objects in the locale of the user running the monitoring application.

1067 **3.7 IANA [and PWG Registration](#) Considerations**

1068 [This MIB does not require any additional registration schemes of IANA, but does depend](#)
1069 [on registration schemes that other Internet standards track specifications have set up. The](#)
1070 [names of these IANA registration assignments under the /in-notes/iana/assignments/ path:](#)

- 1071 1. [printer-language-numbers - used as enums in the documentFormat\(38\)](#)
- 1072 2.

1073 During the development of this standard, the Printer Working Group (PWG) working
1074 with IANA [iana] will register additional enums while the standard is in the proposed and
1075 draft states according to the procedures described in this section. IANA will handle
1076 registration of additional enums after this standard is approved in cooperation with an
1077 IANA-appointed registration editor from the PWG according to the procedures described
1078 in this section:

1079 **3.7.1 IANA Registration of enums**

1080 This specification uses textual conventions to define enumerated values (enums) and bit
1081 values. Enumerations (enums) and bit values are sets of symbolic values defined for use
1082 with one or more objects or attributes. All enumeration sets and bit value sets are
1083 assigned a symbolic data type name (textual convention). As a convention the symbolic
1084 name ends in "TC" for textual convention. These enumerations are defined at the
1085 beginning of the MIB module specification.

1086 This working group has defined several type of enumerations for use in the Job
1087 Monitoring MIB and the Printer MIB[print-mib]. These types differ in the method
1088 employed to control the addition of new enumerations. Throughout this document,
1089 references to "type n enum", where n can be 1, 2 or 3 can be found in the various tables.
1090 The definitions of these types of enumerations are:

1091 3.7.1.1 Type 1 enumerations

1092 Type 1 enumeration: All the values are defined in the Job Monitoring MIB specification
1093 (RFC for the Job Monitoring MIB). Additional enumerated values require a new RFC.

1094 There are no type 1 enums in the current draft.

1095 3.7.1.2 Type 2 enumerations

1096 Type 2 enumeration: An initial set of values are defined in the Job Monitoring MIB
1097 specification. Additional enumerated values are registered after review by this working
1098 group or an editor appointed by IANA after this working group is no longer active.

1099 The following type 2 enums are contained in the current draft :

- 1100 1. **JmUTF8StringTC**

- 1101 2. **JmJobStringTC**
- 1102 3. **JmNaturalLanguageTagTC**
- 1103 4. **JmTimeStampTC**
- 1104 5. **JmFinishingTC** [same enum values as IPP "finishing" attribute]
- 1105 6. **JmPrintQualityTC** [same enum values as IPP "print-quality" attribute]
- 1106 7. **JmTonerEconomyTC**
- 1107 8. **JmMediumTypeTC**
- 1108 9. **JmJobSubmissionIDTypeTC**
- 1109 10. **JmJobCollationTypeTC**
- 1110 11. **JmJobStateTC** [same enum values as IPP "job-state" attribute]
- 1111 12. **JmAttributeTypeTC**

1112 For those textual conventions that have the same enum values as the indicated IPP Job
1113 attribute SHALL be simultaneously registered by IANA for use with IPP [ipp-model] and
1114 the Job Monitoring MIB.

1115 3.7.1.3 Type 3 enumeration

1116 Type 3 enumeration: An initial set of values are defined in the Job Monitoring MIB
1117 specification. Additional enumerated values are registered through IANA without
1118 working group review.

1119 There are no type 3 enums in the current draft.

1120 3.7.2 IANA Registration of type 2 bit values

1121 This draft contains the following type 2 bit value textual-conventions:

- 1122 1. **JmJobServiceTypesTC**
- 1123 2. **JmJobStateReasons1TC**
- 1124 3. **JmJobStateReasons2TC**
- 1125 4. **JmJobStateReasons3TC**
- 1126 5. **JmJobStateReasons4TC**

1127 These textual-conventions are defined as bits in an Integer so that they can be used with
1128 SNMPv1 SMI. The **jobStateReasonsN** ($N=1..4$) attributes are defined as bit values using
1129 the corresponding **JmJobStateReasonsNTC** textual-conventions.

1130 The registration of **JmJobServiceTypesTC** and **JmJobStateReasonsNTC** bit values
1131 SHALL follow the procedures for a type 2 enum as specified in Section 3.7.1.2.

1132 3.7.3 IANA Registration of Job Submission Id Formats

1133 In addition to enums and bit values, this specification assigns a single ASCII digit or
1134 letter to various job submission ID formats. See the **JmJobSubmissionIDTypeTC**
1135 textual-convention and the object. The registration of **jmJobSubmissionID** format
1136 numbers SHALL follow the procedures for a type 2 enum as specified in Section 3.7.1.2.

1137 3.7.4 IANA Registration of MIME types/sub-types for document-formats

1138 The **documentFormat(38)** attribute has MIME type/sub-type values for indicating
1139 document formats which IANA registers as "media type" names. The values of the
1140 **documentFormat(38)** attribute are the same as the corresponding Internet Printing
1141 Protocol (IPP) "document-format" Job attribute values [ipp-model].

1142 3.8 Security Considerations

1143 3.8.1 Read-Write objects

1144 All objects are read-only, greatly simplifying the security considerations. If another MIB
1145 augments this MIB, that MIB might accept SNMP Write operations to objects in that
1146 MIB whose effect is to modify the values of read-only objects in this MIB. However, that
1147 MIB SHALL have to support the required access control in order to achieve security, not
1148 this MIB.

1149 3.8.2 Read-Only Objects In Other User's Jobs

1150 The security policy of some sites MAY be that unprivileged users can only get the objects
1151 from jobs that they submitted, plus a few minimal objects from other jobs, such as the
1152 **jmJobKOctetsPerCopyRequested** and **jmJobKOctetsProcessed** objects, so that a user
1153 can tell how busy a printer is. Other sites MAY allow all unprivileged users to see all
1154 objects of all jobs. This MIB does not require, nor does it specify how, such restrictions
1155 would be implemented. A monitoring application SHOULD enforce the site security
1156 policy with respect to returning information to an unprivileged end user that is using the
1157 monitoring application to monitor jobs that do not belong to that user, i.e., the
1158 **jmJobOwner** object in the **jmJobTable** does not match the user's user name.

1159 An operator is a privileged user that would be able to see all objects of all jobs,
1160 independent of the policy for unprivileged users.

1161 3.9 Notifications

1162 This MIB does not specify any notifications. For simplicity, management applications are
1163 expected to poll for status. The **jmGeneralJobPersistence** and
1164 **jmGeneralAttributePersistence** objects assist an application to determine the polling
1165 rate. The resulting network traffic is not expected to be significant.

1166 4. MIB specification

1167 The following pages constitute the actual Job Monitoring MIB.

```

1168 Job-Monitoring-MIB DEFINITIONS ::= BEGIN
1169
1170 IMPORTS
    MODULE-IDENTITY, OBJECT-TYPE, enterprises, Integer32
    TEXTUAL-CONVENTION
    MODULE-COMPLIANCE, OBJECT-GROUP
    FROM SNMPv2-SMI
    FROM SNMPv2-TC
    FROM SNMPv2-CONF;

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

1171 -- Use the enterprises arc assigned to the PWG which is pwg(2699)
1172 -- and assign the first value: jobmon(1) immediately under pwg(2699).
1173
1174
1175 -- Since this specification was so near to approval by the PWG,
1176 -- no experimental arc has been assigned. In the future, when
1177 -- experimental arcs are needed during the development of
1178 -- other PWG standards (whether SNMP MIBs or other usages
1179 -- for OBJECT IDENTIFIERS), the PWG will assign an experimental arc
1180 -- value that will be distinct from the arc that the PWG assigns when
1181 -- the PWG approves that PWG standard.
1182 -- Thus in the future, experimental and standard arcs will be
1183 -- assigned by the PWG immediately under the pwg(2699) arc.
1184 -- Assign two arcs under that: standard(1) and experimental(2)
1185 -- for all PWG usage.
1186 -- Use the experimental arc until the PWG agrees that the MIB
1187 -- is approved as a PWG standard.
1188
1189 -- Upon publication of the Job Monitoring MIB as a PWG standard
1190 -- and as an Informational RFC, change the second to last arc
1191 -- from experimental(2) to standard(1).
1192 -- This will make it easier to translate prototypes to
1193 -- the standard namespace because the lengths of the OIDs won't
1194 -- change.
1195
1196 jobmonMIB MODULE-IDENTITY
1197     LAST-UPDATED "971211020000Z"
1198     ORGANIZATION "IETF Printer MIB Working Group (PWG)"
1199     CONTACT-INFO
1200         "Tom Hastings
1201         Postal: Xerox Corp.
1202         Mail stop ESAE-231
1203         701 S. Aviation Blvd.
1204         El Segundo, CA 90245"

```

1205
 1206 Tel: (301)333-6413
 1207 Fax: (301)333-5514
 1208 E-mail: hastings@cp10.es.xerox.com
 1209

1210 Send comments to the [Printer Working Group \(PWG\)](#)~~printmib-WG~~
 1211 using the Job Monitoring Project (JMP) Mailing List:

1212 jmp@pwg.org

1213
 1214
 1215 ~~To learn how to subscribe to the JMP mailing list,~~
 1216 ~~send email to: jmp-request@pwg.org~~

1217
 1218 For further information, [including how to subscribe to the](#)
 1219 [jmp mailing list](#), access the PWG web page under 'JMP':
 1220 <http://www.pwg.org/>

1221 DESCRIPTION

1222 "The MIB module for monitoring job in servers, printers, and other devices.

1223
 1224 ~~File: draft-ietf-printmib-job-monitor-07.txt~~

1225 Version: ~~1.00-87~~"

1226 ::= { enterprises pwg(2699) ~~experimental(2)~~ jobmon(1) }

1227
 1228
 1229 -- Textual conventions for this MIB module
 1230
 1231
 1232

1233 1234 **JmUTF8StringTC** ::= TEXTUAL-CONVENTION

1235 DISPLAY-HINT "255a"

1236 STATUS current

1237 DESCRIPTION

1238 "To facilitate internationalization, this TC represents information taken from the ISO/IEC IS
 1239 10646-1 character set, encoded as an octet string using the UTF-8 character encoding scheme."

1240 REFERENCE

1241 "See section 3.6.1, entitled: 'Text generated by the server or device'."

1242 SYNTAX OCTET STRING (SIZE (0..63))
 1243
 1244
 1245

1246 1247 **JmJobStringTC** ::= TEXTUAL-CONVENTION

1248 STATUS current

1249 DESCRIPTION

1250 "To facilitate internationalization, this TC represents information using any coded character set
 1251 registered by IANA as specified in section 0. While it is recommended that the coded character

1252 set be UTF-8 [UTF-8], the actual coded character set SHALL be indicated by the value of the
1253 **jobCodedCharSet(8)** attribute for the job."

1254 REFERENCE

1255 "See section 3.6.2, entitled: 'Text supplied by the job submitter'."

1256 SYNTAX OCTET STRING (SIZE (0..63))

1257

1258

1259

1260

1261 **JmNaturalLanguageTagTC** ::= TEXTUAL-CONVENTION

1262 STATUS current

1263 DESCRIPTION

1264 "An IETF RFC 1766-compliant 'language tag', with zero or more sub-tags that identify a natural
1265 language. While RFC 1766 specifies that the US-ASCII values are case-insensitive, this MIB
1266 specification requires that all characters SHALL be lower case in order to simplify comparing
1267 by management applications."

1268 REFERENCE

1269 "See section 3.6.1, entitled: 'Text generated by the server or device' and section 3.6.2, entitled:

1270 'Text supplied by the job submitter'."

1271 SYNTAX OCTET STRING (SIZE (0..63))

1272

1273

1274

1275 **JmTimeStampTC** ::= TEXTUAL-CONVENTION

1276 STATUS current

1277 DESCRIPTION

1278 "The simple time at which an event took place. The units SHALL be in seconds since the
1279 system was booted.

1280

1281 NOTE - **JmTimeStampTC** is defined in units of seconds, rather than 100ths of seconds, so as
1282 to be simpler for agents to implement (even if they have to implement the 100ths of a second to
1283 comply with implementing **sysUpTime** in MIB-II[mib-II].)

1284

1285 NOTE - **JmTimeStampTC** is defined as an **Integer32** so that it can be used as a value of an
1286 attribute, i.e., as a value of the **jmAttributeValueAsInteger** object. The **TimeStamp** textual-
1287 convention defined in SMNPv2-TC [[SMIv2-TC](#)] is defined as an **APPLICATION 3**
1288 **IMPLICIT INTEGER** tag, not an **Integer32** [which is defined in SNMPv2-SMI \[SMIv2-TC\]](#)
1289 [as UNIVERSAL 2 IMPLICIT INTEGER](#), so cannot be used in this MIB as one of the values of
1290 **jmAttributeValueAsInteger**."

1291 SYNTAX INTEGER(0..2147483647)

1292

1293

1294

1295

1296 **JmJobSourcePlatformTypeTC** ::= TEXTUAL-CONVENTION

1297 STATUS current

1298 DESCRIPTION

1299 "The source platform type that can submit jobs to servers or devices in any of the 3
1300 configurations."
1301 REFERENCE
1302 "This is a type 2 enumeration. See Section 3.7.1.2. [See also IANA operating-system-names](#)
1303 [registry.](#)"
1304 SYNTAX INTEGER {
other(1),
unknown(2),
sptUNIX(3), -- UNIX
sptOS2(4), -- OS/2
sptPCDOS(5), -- DOS
sptNT(6), -- NT
sptMVS(7), -- MVS
sptVM(8), -- VM
sptOS400(9), -- OS/400
sptVMS(10), -- VMS
sptWindows(11), -- Windows
sptNetWare(12) -- NetWare
1305 }
1306
1307
1308
1309
1310
1311 **JmFinishingTC** ::= TEXTUAL-CONVENTION
1312 STATUS current
1313 DESCRIPTION
1314 "The type of finishing operation.
1315
1316 These values are the same as the enum values of the IPP 'finishings' attribute. See Section
1317 3.7.1.2.
1318
1319 **other(1),**
1320 Some other finishing operation besides one of the specified or registered values.
1321
1322 **unknown(2),**
1323 The finishing is unknown.
1324
1325 **none(3),**
1326 Perform no finishing.
1327
1328 **staple(4),**
1329 Bind the document(s) with one or more staples. The exact number and placement of the
1330 staples is site-defined.
1331
1332 **punch(5),**
1333 This value indicates that holes are required in the finished document. The exact number
1334 and placement of the holes is site-defined. The punch specification MAY be satisfied (in

1335 a site- and implementation-specific manner) either by drilling/punching, or by
 1336 substituting pre-drilled media.
 1337
 1338 **cover(6),**
 1339 This value is specified when it is desired to select a non-printed (or pre-printed) cover for
 1340 the document. This does not supplant the specification of a printed cover (on cover stock
 1341 medium) by the document itself.
 1342
 1343 **bind(7)**
 1344 This value indicates that a binding is to be applied to the document; the type and
 1345 placement of the binding is product-specific."
 1346 REFERENCE
 1347 "This is a type 2 enumeration. See Section 3.7.1.2."
 1348 SYNTAX INTEGER {
 1349 other(1),
 1350 unknown(2),
 1351 none(3),
 1352 staple(4),
 1353 punch(5),
 1354 cover(6),
 1355 bind(7)
 1356 }
 1357
 1358
 1359
 1360
 1361
 1362 **JmPrintQualityTC ::= TEXTUAL-CONVENTION**
 1363 STATUS current
 1364 DESCRIPTION
 1365 "Print quality settings.
 1366
 1367 These values are the same as the enum values of the IPP 'print-quality' attribute. See Section
 1368 3.7.1.2."
 1369 REFERENCE
 1370 "This is a type 2 enumeration. See Section 3.7.1.2."
 1371 SYNTAX INTEGER {
 1372 **other(1),** -- Not one of the specified or registered values.
 1373 --
 1374 **unknown(2),** -- The actual value is unknown.
 1375 **draft(3),** -- Lowest quality available on the printer.
 1376 **normal(4),** -- Normal or intermediate quality on the printer.
 1377 --
 1378 **high(5)** -- Highest quality available on the printer.
 1379 }
 1380
 1381
 1382
 1383

1376

1377 **JmPrinterResolutionTC** ::= TEXTUAL-CONVENTION

1378 STATUS current

1379 DESCRIPTION

1380 "Printer resolutions.

1381

1382 Nine octets consisting of two 4-octet SIGNED-INTEGERS followed by a SIGNED-BYTE. The

1383 values are the same as those specified in the Printer MIB [printmib]. The first SIGNED-

1384 INTEGER contains the value of prtMarkerAddressabilityXFeedDir. The second SIGNED-

1385 INTEGER contains the value of prtMarkerAddressabilityFeedDir. The SIGNED-BYTE

1386 contains the value of prtMarkerAddressabilityUnit.

1387

1388 Note: the latter value is either 3 (tenThousandsOfInches) or 4 (micrometers) and the

1389 addressability is in 10,000 units of measure. Thus the SIGNED-INTEGERS represent integral

1390 values in either dots-per-inch or dots-per-centimeter.

1391

1392 The syntax is the same as the IPP 'printer-resolution' attribute. See Section 3.7.1.2."

1393 SYNTAX OCTET STRING (SIZE(9))

1394

1395

1396

1397

1398

1399 **JmTonerEconomyTC** ::= TEXTUAL-CONVENTION

1400 STATUS current

1401 DESCRIPTION

1402 "Toner economy settings."

1403 REFERENCE

1404 "This is a type 2 enumeration. See Section 3.7.1.2."

1405 SYNTAX INTEGER {

unknown(2), -- unknown.

off(3), -- Off. Normal. Use full toner.

on(4) -- On. Use less toner than normal.

1406 }

1407

1408

1409

1410

1411

1412 **JmBooleanTC** ::= TEXTUAL-CONVENTION

1413 STATUS current

1414 DESCRIPTION

1415 "Boolean true or false value."

1416 REFERENCE

1417 "This is a type 2 enumeration. See Section 3.7.1.2."

1418 SYNTAX INTEGER {

unknown(2), -- unknown.

```
1419         false(3),           -- FALSE.
1420         true(4)             -- TRUE.
1421     }
1422
1423
1424
1425 JmMediumTypeTC ::= TEXTUAL-CONVENTION
1426     STATUS      current
1427     DESCRIPTION
1428         "Identifies the type of medium.
1429
1430         other(1),
1431             The type is neither one of the values listed in this specification nor a registered value.
1432
1433         unknown(2),
1434             The type is not known.
1435
1436         stationery(3),
1437             Separately cut sheets of an opaque material.
1438
1439         transparency(4),
1440             Separately cut sheets of a transparent material.
1441
1442         envelope(5),
1443             Envelopes that can be used for conventional mailing purposes.
1444
1445         envelopePlain(6),
1446             Envelopes that are not preprinted and have no windows.
1447
1448         envelopeWindow(7),
1449             Envelopes that have windows for addressing purposes.
1450
1451         continuousLong(8),
1452             Continuously connected sheets of an opaque material connected along the long edge.
1453
1454         continuousShort(9),
1455             Continuously connected sheets of an opaque material connected along the short edge.
1456
1457         tabStock(10),
1458             Media with tabs.
1459
1460         multiPartForm(11),
1461             Form medium composed of multiple layers not pre-attached to one another; each sheet
1462             MAY be drawn separately from an input source.
1463
```

1464 **labels(12),**
 1465 Label-stock.

1466

1467 **multiLayer(13)**
 1468 Form medium composed of multiple layers which are pre-attached to one another, e.g. for
 1469 use with impact printers."

1470 REFERENCE
 1471 "This is a type 2 enumeration. See Section 3.7.1.2. [These enum values correspond to the](#)
 1472 [keyword name strings of the prtInputMediaType object in the Printer MIB \[print-mib\]. There](#)
 1473 [is no printer description attribute in IPP/1.0 that represents these values.](#)"

1474 SYNTAX INTEGER {
 1475 other(1),
 1476 unknown(2),
 1477 stationery(3),
 1478 transparency(4),
 1479 envelope(5),
 1480 envelopePlain(6),
 1481 envelopeWindow(7),
 1482 continuousLong(8),
 1483 continuousShort(9),
 1484 tabStock(10),
 1485 multiPartForm(11),
 1486 labels(12),
 1487 multiLayer(13)
 1488 }
 1489
 1490
 1491
 1492
 1493

1494 **JmJobCollationTypeTC ::= TEXTUAL-CONVENTION**
 1495 STATUS current
 1496 DESCRIPTION
 1497 "This value is the type of ~~job sheet and document~~ collation. [Implementations that don't support](#)
 1498 [multiple documents or don't support multiple copies SHALL NOT support the](#)
 1499 [uncollatedDocuments\(5\) value.](#)

1500

1501 **other(1);**
 1502 Some other collation besides one of the specified or registered values.

1503

1504 **unknown(2);**
 1505 The collation is unknown.

1506

1507 **externalSheetCollation(3);**
 1508 Collation of the sheets within a document copy is performed externally to the printing
 1509 device, either in an attached physical output bin collator or is uncollated (so that the user
 1510 does the sheet collation by hand).

1511

1512 Note that uncollated and collation to a series of output bins are the same in terms of the
 1513 behavior of the job MIB Impression and Sheet completed attributes. Therefore, we call
 1514 this form External Sheet Collation.

1515
 1516 **internalSheetCollationWithCollatedDoes(4),**

1517 Collation of the sheets within each document copy is performed within the printing
 1518 device by making multiple passes over either the source or an intermediate representation
 1519 of the document. In addition, when there are multiple documents per job, the i'th copy of
 1520 each document is stacked before the j'th copy of each document, i.e., the documents are
 1521 collated within each job copy.

1522
 1523 If **jobCopiesRequested** or **documentCopiesRequested** = 1, then **collationType** is
 1524 defined as 4.

1525
 1526 **internalSheetCollationWithUncollatedDoes(5),**

1527 Collation of the sheets within each document copy is performed within the printing
 1528 device by making multiple passes over either the source or an intermediate representation
 1529 of the document. In addition, when there are multiple documents per job, all copies of
 1530 the first document in the job are stacked before the any copied of the next document in
 1531 the job, i.e., the documents are uncollated within the job.

1532
 1533 REFERENCE

1534 "This is a type 2 enumeration. See Section 3.7.1.2. [See the definitions of the terms: 'job](#)
 1535 [collation', 'collated documents', 'uncollated documents', and 'uncollated sheets' in the](#)
 1536 [terminology section, 2. See also Section 3.4, entitled 'Monitoring Job Progress'.](#)"

1537 SYNTAX INTEGER {

1538 other(1),

1539 unknown(2),

1540 [uncollatedSheetexternalSheetCollation\(3\)](#), -- Uncollated Sheets

1541 [internalSheetCollationWithCollatedDocuments\(4\)](#), -- Collated Documents

1542 [internalSheetCollationWithUncollatedDocuments\(5\)](#) -- Uncollated Documents

1543 }

1544
 1545
 1546
 1547 **JmJobSubmissionIDTypeIDTC** ::= TEXTUAL-CONVENTION

1548 STATUS current

1549 DESCRIPTION

1550 "Identifies the format type of a job submission ID.

1551
 1552 Each job submission ID is a fixed-length, 48-octet printable US-ASCII [US-ASCII] coded
 1553 character string containing no control characters, consisting of the following fields:

1554
 1555 octet 1: The format letter identifying the format. The US-ASCII characters '0-9', 'A-Z', and
 1556 'a-z' are assigned in order giving 62 possible formats.

1557 octets 2-40: A 39-character, US-ASCII trailing SPACE filled field specified by the format
 1558 letter, if the data is less than 39 ASCII characters.

1559 octets 41-48: A sequential or random [US-ASCII](#) number to make the ID quasi-unique.

1560

1561 If the client does not supply a job submission ID in the job submission protocol, then the agent
 1562 SHALL assign a job submission ID using any of the standard formats that are reserved for the
 1563 agent. Clients SHALL not use formats that are reserved for agents and agents SHALL NOT use
 1564 formats that are reserved for clients, in order to reduce conflicts in ID generation. See the
 1565 description for which formats are reserved for clients or for agents.
 1566

1567 Registration of additional formats may be done following the procedures described in Section
 1568 3.7.3.

1570 The format values defined at the time of completion of this specification are:

1572 Format

1573 Letter Description

1574 -----

1575 **'0'** [Job Owner generated by the server/device](#)

1576 octets 2-40: [The](#) last 39 bytes of the **jmJobOwner** object.

1577 octets 41-48: [The US-ASCII](#) 8-decimal-digit sequential number [assigned by the agent](#).

1578 This format is reserved for agents.
 1579

1580 NOTE - Clients wishing to use a job submission ID that incorporates the job owner, SHALL
 1581 use format '8', not format '0'.
 1582

1583 **'1'** [Job Name](#)

1584 octets 2-40: [The](#) last 39 bytes of the **jobName** attribute.

1585 octets 41-48: [The US-ASCII](#) 8-decimal-digit random number [assigned by the client](#).

1586 This format is reserved for clients.
 1587

1588 **'2'** [Client MAC address](#)

1589 octets 2-40: [The](#) client MAC address: in hexadecimal with each nibble of the 6 octet address
 1590 being '0'-'9' or 'A' - 'F' (uppercase only). Most significant octet first.

1591 octets 41-48: [The US-ASCII](#) 8-decimal-digit sequential number [assigned by the client](#).

1592 This format is reserved for clients.
 1593

1594 **'3'** [Client URL](#)

1595 octets 2-40: [The](#) last 39 bytes of the client URL [URI-spec].

1596 octets 41-48: [The US-ASCII](#) 8-decimal-digit sequential number [assigned by the client](#).

1597 This format is reserved for clients.
 1598

1599 **'4'** [Job URI](#)

1600 octets 2-40: [The](#) last 39 bytes of the URI [URI-spec] assigned by the server or device to the job
 1601 when the job was submitted for processing.

1602 octets 41-48: [The US-ASCII](#) 8-decimal-digit sequential number [assigned by the agent](#).

1603 This format is reserved for agents.
 1604

1605 **'5'** [POSIX User Number](#)

1606 octets 2-40: [The](#) last 39 bytes of a user number, such as POSIX user number.

1607 octets 41-48: [The US-ASCII](#) 8-decimal-digit sequential number [assigned by the client](#).

1608 This format is reserved for clients.
 1609

- 1610 **'6'** [User Account Number](#)
1611 octets 2-40: [The](#) last 39 bytes of the user account number.
1612 octets 41-48: [The US-ASCII](#) 8-decimal-digit sequential number [assigned by the client](#).
1613 This format is reserved for clients.
1614
- 1615 **'7'** [DTMF Incoming FAX routing number](#)
1616 octets 2-40: [The](#) last 39 bytes of the DTMF incoming FAX routing number.
1617 octets 41-48: [The US-ASCII](#) 8-decimal-digit sequential number [assigned by the client](#).
1618 This format is reserved for clients.
1619
- 1620 **'8'** [Job Owner supplied by the client](#)
1621 octets 2-40: [The](#) last 39 bytes of the job owner name (that the agent returns in the
1622 **jmJobOwner** object).
1623 octets 41-48: [The US-ASCII](#) 8-decimal-digit sequential number [assigned by the client](#).
1624 This format is reserved for clients. [See format '0' which is reserved for agents](#).
1625
- 1626 **'9'** [Host Name](#)
1627 octets 2-40: [The](#) last 39 bytes of the host name with trailing SPACES that submitted the job to
1628 this server/device using a protocol, such as LPD [RFC-1179] which includes the host
1629 name in the job submission protocol.
1630 octets 41-48: [The US-ASCII](#) 8-decimal-digit leading zero representation of the job id generated
1631 [by the](#) by the submitting server (configuration 3) or the client (configuration 1 and 2),
1632 such as in the LPD protocol.
1633 This format is reserved for clients.
1634
- 1635 **'A'** [AppleTalk Protocol](#)
1636 octets 2-40: [Contains the AppleTalk printer name, with the first character of the name in octet](#)
1637 [2. AppleTalk printer names are a maximum of 31 characters. Any unused portion of this](#)
1638 [field shall be filled with spaces.](#)
1639 octets 41-48: ['00000XXX'](#), where 'XXX' is the 3-digit US-ASCII decimal representation of the
1640 [Connection Id.](#)
1641 [This format is reserved for agents.](#)
1642
- 1643 **'B'** [NetWare PServer](#)
1644 octets 2-40: [Contains the Directory Path Name as recorded by the Novell File Server in the](#)
1645 [queue directory. If the string is less than 40 octets, the left-most character in the string](#)
1646 [shall appear in octet position 2. Otherwise, only the last 39 bytes shall be included. Any](#)
1647 [unused portion of this field shall be filled with spaces.](#)
1648 octets 41-48: ['000XXXXX'](#) The US-ASCII representation of the Job Number as per the
1649 [NetWare File Server Queue Management Services.](#)
1650 [This format is reserved for agents.](#)
1651
- 1652 **'C'** [Server Message Block protocol \(SMB\)](#)
1653 octets 2-40: [Contains a decimal \(US-ASCII coded\) representation of the 16 bit SMB Tree Id](#)
1654 [field, which uniquely identifies the connection that submitted the job to the printer. The](#)
1655 [most significant digit of the numeric string shall be placed in octet position 2. All unused](#)
1656 [portions of this field shall be filled with spaces. The SMB Tree Id has a maximum value](#)
1657 [of 65,535.](#)

1658 [octets 41-48: The US-ASCII 8-decimal-digit leading zero representation of the File Handle](#)
 1659 [returned from the device to the client in response to a Create Print File command.](#)
 1660 [This format is reserved for agents.](#)
 1661
 1662 ['D' Transport Independent Printer/System Interface \(TIP/SI\)](#)
 1663 [octets 2-40: Contains the Job Name from the Job Control-Start Job \(JC-SJ\) command. If the](#)
 1664 [Job Name portion is less than 40 octets, the left-most character in the string shall appear](#)
 1665 [in octet position 2. Any unused portion of this field shall be filled with spaces.](#)
 1666 [Otherwise, only the last 39 bytes shall be included.](#)
 1667 [octets 41-48: The US-ASCII 8-decimal-digit leading zero representation of the **jmJobIndex**](#)
 1668 [assigned by the agent.](#)
 1669 [This format is reserved for agents, since the agent supplies octets 41-48, though the client](#)
 1670 [supplies the job name. See format 'I' reserved to clients to submit job name ids in which](#)
 1671 [they supply octets 41-48.](#)
 1672

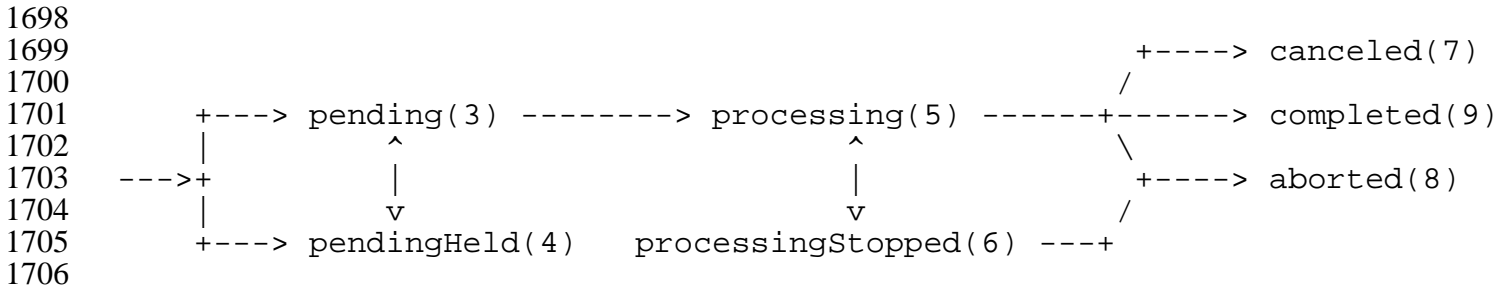
1673 NOTE - the job submission id is only intended to be unique between a limited set of clients for
 1674 a limited duration of time, namely, for the life time of the job in the context of the server or
 1675 device that is processing the job. Some of the formats include something that is unique per
 1676 client and a random number so that the same job submitted by the same client will have a
 1677 different job submission id. For other formats, where part of the id is guaranteed to be unique
 1678 for each client, such as the MAC address or URL, a sequential number SHOULD suffice for
 1679 each client (and may be easier for each client to manage). Therefore, the length of the job
 1680 submission id has been selected to reduce the probability of collision to an extremely low
 1681 number, but is not intended to be an absolute guarantee of uniqueness. None-the-less,
 1682 collisions are remotely possible, but without bad consequences, since this MIB is intended to be
 1683 used only for monitoring jobs, not for controlling and managing them."

1684 REFERENCE

1685 "This is like a type 2 enumeration. See section 3.7.3."

1686 SYNTAX OCTET STRING(SIZE(1)) -- ASCII '0'-'9', 'A'-'Z', 'a'-'z'

1691
 1692 **JmJobStateTC ::= TEXTUAL-CONVENTION**
 1693 STATUS current
 1694 DESCRIPTION
 1695 "The current state of the job (**pending, processing, completed**, etc.).
 1696
 1697 The following figure shows the normal job state transitions:



1707 **Figure 4 - Normal Job State Transitions**

1708
 1709 Normally a job progresses from left to right. Other state transitions are unlikely, but are not
 1710 forbidden. Not shown are the transitions to the **canceled** state from the **pending**,
 1711 **pendingHeld**, and **processingStopped** states.

1712
 1713 Jobs in the **pending**, **processing**, and **processingStopped** states are called 'active', while jobs in
 1714 the **pendingHeld**, **canceled**, **aborted**, and **completed** states are called 'inactive'. Jobs reach
 1715 one of the three terminal states: **completed**, **canceled**, or **aborted**, *after* the jobs have
 1716 completed all activity, and all MIB objects and attributes have reached their final values for the
 1717 job.

1718
 1719 These values are the same as the enum values of the IPP 'job-state' job attribute. See Section
 1720 3.7.1.2.

1721
 1722 **unknown(2),**
 1723 The job state is *not* known, or its state is indeterminate.

1724
 1725 **pending(3),**
 1726 The job is a candidate to start processing, but is not yet processing.

1727
 1728 **pendingHeld(4),**
 1729 The job is not a candidate for processing for any number of reasons but will return to the
 1730 **pending** state as soon as the reasons are no longer present. The job's
 1731 **jmJobStateReasons1** object and/or **jobStateReasonsN** ($N=2..4$) attributes SHALL
 1732 indicate why the job is no longer a candidate for processing. The reasons are represented
 1733 as bits in the **jmJobStateReasons1** object and/or **jobStateReasonsN** ($N=2..4$) attributes.
 1734 See the **JmJobStateReasonsVTC** ($N=1..4$) textual convention for the specification of
 1735 each reason.

1736
 1737 **processing(5),**
 1738 One or more of:
 1739
 1740 1. the job is using, or is attempting to use, one or more purely software processes that are
 1741 analyzing, creating, or interpreting a PDL, etc.,
 1742
 1743 2. the job is using, or is attempting to use, one or more hardware devices that are
 1744 interpreting a PDL, making marks on a medium, and/or performing finishing, such as
 1745 stapling, etc.,

1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794

OR

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

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

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

processingStopped(6),

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

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

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

canceled(7),

A client has canceled the job and the server or device has completed canceling the job AND all MIB objects and attributes have reached their final values for the job. While the server or device is canceling the job, the job's **jmJobStateReasons1** object SHOULD contain the **processingToStopPoint** value and one of the **canceledByUser**, **canceledByOperator**, or **canceledAtDevice** values. The **canceledByUser**, **canceledByOperator**, or **canceledAtDevice** values remain while the job is in the **canceled** state.

aborted(8),

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

1795 **completed(9)**
 1796 The job has completed successfully or with warnings or errors after processing and all of
 1797 the media have been successfully stacked in the appropriate output bin(s) *AND* all MIB
 1798 objects and attributes have reached their final values for the job. The job's
 1799 **jmJobStateReasons1** object SHOULD contain one of: **completedSuccessfully**,
 1800 **completedWithWarnings**, or **completedWithErrors** values."

1801 REFERENCE
 1802 "This is a type 2 enumeration. See Section 3.7.1.2."
 1803 SYNTAX INTEGER {
 1804 unknown(2),
 1805 pending(3),
 1806 pendingHeld(4),
 1807 processing(5),
 1808 processingStopped(6),
 1809 canceled(7),
 1810 aborted(8),
 1811 completed(9)
 1812 }
 1813

1814

1815 **JmAttributeTypeTC ::= TEXTUAL-CONVENTION**
 1816 STATUS current
 1817 DESCRIPTION
 1818 "The type of the attribute which identifies the attribute."
 1819

1820 In the following definitions of the enums, each description indicates whether the useful value of
 1821 the attribute SHALL be represented using the **jmAttributeValueAsInteger** or the
 1822 **jmAttributeValueAsOctets** objects by the initial tag: **INTEGER:**' or **OCTETS:**',
 1823 respectively.
 1824

1825 Some attributes allow the agent implementer a choice of useful values of either an integer, an
 1826 octets representation, or both, depending on implementation. These attributes are indicated
 1827 with **INTEGER:**' AND/OR **OCTETS:**' tags.
 1828

1829 A very few attributes require both objects at the same time to represent a pair of useful values
 1830 (see **mediumConsumed(171)**). These attributes are indicated with **INTEGER:**' AND
 1831 **OCTETS:**' tags. See the **jmAttributeGroup** for the descriptions of these two MANDATORY
 1832 objects.
 1833

1834 NOTE - The enum assignments are grouped logically with values assigned in groups of 20, so
 1835 that additional values may be registered in the future and assigned a value that is part of their
 1836 logical grouping.
 1837

1838 Values in the range 2**30 to 2**31-1 are reserved for private or experimental usage. This
 1839 range corresponds to the same range reserved in IPP. Implementers are warned that use of such
 1840 values may conflict with other implementations. Implementers are encouraged to request
 1841 registration of enum values following the procedures in Section 3.7.1.
 1842

1843 NOTE: No attribute name exceeds 31 characters.

1844
1845 The standard attribute types defined at the time of completion of the specification are:

1846 jmAttributeTypeIndex	Datatype
1847 -----	-----
1848	
1849	
1850 other(1),	Integer32(-2..2147483647)
1851	AND/OR
1852	OCTET STRING(SIZE(0..63))
1853	INTEGER: and/or OCTETS: An attribute that is not in the list and/or that has not been
1854	approved and registered with IANA.

1855
1856
1857 ++++++

1858 + **Job State attributes**

1859 +
1860 + **The following attributes specify the state of a job.**

1861 ++++++

1862	
1863 jobStateReasons2(3),	JmJobStateReasons2TC
1864	INTEGER: Additional information about the job's current state that augments the
1865	jmJobState object. See the description under the JmJobStateReasons1TC textual-
1866	convention.

1867	
1868 jobStateReasons3(4),	JmJobStateReasons3TC
1869	INTEGER: Additional information about the job's current state that augments the
1870	jmJobState object. See the description under JmJobStateReasons1TC textual-
1871	convention.

1872	
1873 jobStateReasons4(5),	JmJobStateReasons4TC
1874	INTEGER: Additional information about the job's current state that augments the
1875	jmJobState object. See the description under JmJobStateReasons1TC textual-
1876	convention.

1877	
1878 processingMessage(6),	JmUTF8StringTC(SIZE(0..63))
1879	OCTETS: MULTI-ROW: A coded character set message that is generated by the server
1880	or device during the processing of the job as a simple form of processing log to show
1881	progress and any problems. The natural language of each value is specified by the
1882	corresponding processingMessageNaturalLanguageTag(7) value.

1883
1884 NOTE - This attribute is intended for such conditions as interpreter messages, rather than
1885 being the printable form of the **jmJobState** and **jmJobStateReasons1** objects and
1886 **jobStateReasons2**, **jobStateReasons3**, and **jobStateReasons4** attributes. In order to
1887 produce a localized printable form of these job state objects/attribute, a management
1888 application SHOULD produce a message from their enum and bit values.

1889
1890 NOTE - There is no job description attribute in IPP/1.0 that corresponds to this attribute
1891 and this attribute does not correspond to the IPP/1.0 'job-state-message' job description

1892 [attribute, which is just a printable form of the IPP 'job-state' and 'job-state-reasons' job](#)
 1893 [attributes.](#)

1894
 1895 There is no restriction for the same message occurring in multiple rows.
 1896

1897 **processingMessageNaturalLanguageTag(7),** OCTET STRING(SIZE(02..63))
 1898 OCTETS: MULTI-ROW: The natural language of the corresponding
 1899 **processingMessage(6)** attribute [value](#). See section 3.6.1, entitled 'Text generated by the
 1900 server or device'.
 1901

1902 If the agent does not know the natural language of the job processing message, the agent
 1903 SHALL either (1) return a zero length string value for the
 1904 **processingMessageNaturalLanguageTag(7)** attribute or (2) not return the
 1905 **processingMessageNaturalLanguageTag(7)** attribute for the job.
 1906

1907 There is no restriction for the same tag occurring in multiple rows, [since when this](#)
 1908 [attribute is implemented, it SHOULD have a value row for each corresponding](#)
 1909 **processingMessage(6)** [attribute value row](#).
 1910

1911 **jobCodedCharSet(8),** CodedCharSet

1912 INTEGER: The MIBenum identifier of the coded character set that the agent is using to
 1913 represent coded character set objects and attributes of type **JmJobStringTC**'. These
 1914 coded character set objects and attributes are either: (1) supplied by the job submitting
 1915 client or (2) defaulted by the server or device when omitted by the job submitting client.
 1916 The agent SHALL represent these objects and attributes in the MIB either (1) in the coded
 1917 character set as they were submitted or (2) MAY convert the coded character set to
 1918 another coded character set or encoding scheme as identified by the
 1919 **jobCodedCharSet(8)** attribute. See section 3.6.2, entitled 'Text supplied by the job
 1920 submitter'.
 1921

1922 These MIBenum values are assigned by IANA [IANA-charsets] when the coded character
 1923 sets are registered. The coded character set SHALL be one of the ones registered with
 1924 IANA [IANA] and the enum value uses the **CodedCharSet** textual-convention from the
 1925 Printer MIB. See the **JmJobStringTC** textual-convention.
 1926

1927 If the agent does not know what coded character set was used by the job submitting
 1928 client, the agent SHALL either (1) return the **unknown(2)**' value for the
 1929 **jobCodedCharSet(8)** attribute or (2) not return the **jobCodedCharSet(8)** attribute for
 1930 the job.
 1931

1932 **jobNaturalLanguageTag(9),** OCTET STRING(SIZE(02..63))

1933 OCTETS: The natural language of the job attributes supplied by the job submitter or
 1934 defaulted by the server or device for the job, i.e., all objects and attributes represented by
 1935 the '**JmJobStringTC**' textual-convention, such as **jobName**, **mediumRequested**, etc.
 1936 See Section 3.6.2, entitled 'Text supplied by the job submitter'.
 1937

1938 If the agent does not know what natural language was used by the job submitting client,
 1939 the agent SHALL either (1) return a zero length string value for the

1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986

jobNaturalLanguageTag(9) attribute or (2) not return **jobNaturalLanguageTag(9)** attribute for the job.

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

jobURI(20), **OCTET STRING(SIZE(0..63))**
OCTETS: MULTI-ROW: The job's Universal Resource Identifier (URI) [RFC-1738]. See IPP [ipp-model] for example usage.

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.

1987 This attribute is intended for enabling a user or the user's application to convey a job
 1988 name that MAY be printed on a start sheet, returned in a **query** result, or used in
 1989 notification or logging messages.

1990
 1991 In order to assist users to find their jobs for job submission protocols that don't supply a
 1992 **jmJobSubmissionID**, the agent SHOULD maintain the **jobName** attribute for the time
 1993 specified by the **jmGeneralJobPersistence** object, rather than the (shorter)
 1994 **jmGeneralAttributePersistence** object.

1995
 1996 If this attribute is not specified when the job is submitted, no job name is assumed, but
 1997 implementation specific defaults are allowed, such as the value of the **documentName**
 1998 attribute of the first document in the job or the **fileName** attribute of the first document in
 1999 the job.

2000
 2001 The **jobName** attribute is distinguished from the **jobComment** attribute, in that the
 2002 **jobName** attribute is intended to permit the submitting user to distinguish between
 2003 different jobs that he/she has submitted. The **jobComment** attribute is intended to be
 2004 free form additional information that a user might wish to use to communicate with
 2005 himself/herself, such as a reminder of what to do with the results or to indicate a different
 2006 set of input parameters were tried in several different job submissions.

jobServiceTypes(24),**JmJobServiceTypesTC**

2009 INTEGER: Specifies the type(s) of service to which the job has been submitted (print,
 2010 fax, scan, etc.). The service type is bit encoded with each job service type so that more
 2011 general and arbitrary services can be created, such as services with more than one
 2012 destination type, or ones with only a source or only a destination. For example, a job
 2013 service might **scan**, **faxOut**, and **print** a single job. In this case, three bits would be set in
 2014 the **jobServiceTypes** attribute, corresponding to the hexadecimal values: **0x8** + **0x20** +
 2015 **0x4**, respectively, yielding: **0x2C**.

2016
 2017 Whether this attribute is set from a job attribute supplied by the job submission client or
 2018 is set by the recipient job submission server or device depends on the job submission
 2019 protocol. This attribute SHALL be implemented if the server or device has other types in
 2020 addition to or instead of printing.

2021
 2022 One of the purposes of this attribute is to permit a requester to filter out jobs that are not
 2023 of interest. For example, a printer operator may only be interested in jobs that include
 2024 printing.

jobSourceChannelIndex(25),

Integer32(0..2147483647)

2027 INTEGER: The index of the row in the associated Printer MIB[print-mib] of the channel
 2028 which is the source of the print job.

jobSourcePlatformType(26),**JmJobSourcePlatformTypeTC**

2031 INTEGER: The source platform type of the immediate upstream submitter that submitted
 2032 the job to the server (configuration 2) or device (configuration 1 and 3) to which the agent
 2033 is providing access. For configuration 1, this is the type of the client that submitted the
 2034 job to the device; for configuration 2, this is the type of the client that submitted the job

2035 to the server; and for configuration 3, this is the type of the server that submitted the job
 2036 to the device.

2037

2038 **submittingServerName(27),** **JmJobStringTC(SIZE(0..63))**
 2039 OCTETS: For configuration 3 only: The administrative name of the server that
 2040 submitted the job to the device.

2041

2042 **submittingApplicationName(28),** **JmJobStringTC(SIZE(0..63))**
 2043 OCTETS: The name of the client application (not the server in configuration 3) that
 2044 submitted the job to the server or device.

2045

2046 **jobOriginatingHost(29),** **JmJobStringTC(SIZE(0..63))**
 2047 OCTETS: The name of the client host (not the server host name in configuration 3) that
 2048 submitted the job to the server or device.

2049

2050 **deviceNameRequested(30),** **JmJobStringTC(SIZE(0..63))**
 2051 OCTETS: The administratively defined coded character set name of the target device
 2052 requested by the submitting user. For configuration 1, its value corresponds to the Printer
 2053 MIB[print-mib]: **prtGeneralPrinterName** object. For configuration 2 and 3, its value is
 2054 the name of the logical or physical device that the user supplied to indicate to the server
 2055 on which device(s) they wanted the job to be processed.

2056

2057 **queueNameRequested(31),** **JmJobStringTC(SIZE(0..63))**
 2058 OCTETS: The administratively defined coded character set name of the target queue
 2059 requested by the submitting user. For configuration 1, its value corresponds to the queue
 2060 in the device for which the agent is providing access. For configuration 2 and 3, its value
 2061 is the name of the queue that the user supplied to indicate to the server on which device(s)
 2062 they wanted the job to be processed.

2063

2064 NOTE - typically an implementation SHOULD support either the **deviceNameRequested**
 2065 or **queueNameRequested** attribute, but not both.

2066

2067 **physicalDevice(32),** **hrDeviceIndex**
 2068 AND/OR
 2069 **JmUTF8StringTC(SIZE(0..63))**
 2070 INTEGER: MULTI-ROW: The index of the physical device MIB instance
 2071 requested/used, such as the Printer MIB[print-mib]. This value is an **hrDeviceIndex**
 2072 value. See the Host Resources MIB[hr-mib].

2073

2074 AND/OR

2075

2076 OCTETS: MULTI-ROW: The name of the physical device to which the job is assigned.

2077

2078 **numberOfDocuments(33),** **Integer32(-2..2147483647)**
 2079 INTEGER: The number of documents in this job.

2080

2081 The agent SHOULD return this attribute if the job has more than one document.

2082

2083 **fileName(34),** **JmJobStringTC(SIZE(0..63))**
2084 OCTETS: MULTI-ROW: The coded character set file name or URI[URI-spec] of the
2085 document.
2086
2087 There is no restriction on the same file name occurring in multiple rows.
2088
2089 **documentName(35),** **JmJobStringTC(SIZE(0..63))**
2090 OCTETS: MULTI-ROW: The coded character set name of the document.
2091
2092 There is no restriction on the same document name occurring in multiple rows.
2093
2094 **jobComment(36),** **JmJobStringTC(SIZE(0..63))**
2095 OCTETS: An arbitrary human-readable coded character text string supplied by the
2096 submitting user or the job submitting application program for any purpose. For example,
2097 a user might indicate what he/she is going to do with the printed output or the job
2098 submitting application program might indicate how the document was produced.
2099
2100 The **jobComment** attribute is not intended to be a name; see the **jobName** attribute.
2101
2102 **documentFormatIndex(37),** **Integer32(0..2147483647)**
2103 INTEGER: MULTI-ROW: The index in the **prtInterpreterTable** in the Printer
2104 MIB[print-mib] of the page description language (PDL) or control language interpreter
2105 that this job requires/uses. A document or a job MAY use more than one PDL or control
2106 language.
2107
2108 NOTE - As with all intensive attributes where multiple rows are allowed, there SHALL
2109 be only one distinct row for each distinct interpreter; there SHALL be no duplicates.
2110
2111 NOTE - This attribute type is intended to be used with an agent that implements the
2112 Printer MIB and SHALL not be used if the agent does not implement the Printer MIB.
2113 Such an agent SHALL use the **documentFormat** attribute instead.
2114
2115 **documentFormat(38),** **PrtInterpreterLangFamilyTC**
2116 **AND/OR**
2117 **OCTET STRING(SIZE(0..63))**
2118 INTEGER: MULTI-ROW: The interpreter language family corresponding to the Printer
2119 MIB[print-mib] **prtInterpreterLangFamily** object, that this job requires/uses. A
2120 document or a job MAY use more than one PDL or control language.
2121
2122 AND/OR
2123
2124 OCTETS: MULTI-ROW: The document format registered as a media type[iana-media-
2125 types], i.e., the name of the MIME content-type/subtype. Examples:
2126 'application/postscript', 'application/vnd.hp-PCL', 'application/pdf', 'text/plain' (US-ASCII
2127 SHALL be assumed), 'text/plain; charset=iso-8859-1', and 'application/octet-stream'. [The](#)
2128 [IPP 'document-format' job attribute uses these same values with the same semantics.](#) See
2129 the IPP [ipp-model] 'mimeMediaType' attribute syntax and the "document-format"
2130 attribute for further examples and explanation.
2131

2132
 2133
 2134 + Job Parameter attributes
 2135 +
 2136 + The following attributes represent input parameters
 2137 + supplied by the submitting client in the job submission
 2138 + protocol.
 2139 +
 2140
 2141 **jobPriority(50), Integer32(-21..100)**
 2142 INTEGER: The priority for scheduling the job. It is used by servers and devices that
 2143 employ a priority-based scheduling algorithm.
 2144
 2145 A higher value specifies a higher priority. The value **1** is defined to indicate the lowest
 2146 possible priority (a job which a priority-based scheduling algorithm SHALL pass over in
 2147 favor of higher priority jobs). The value **100** is defined to indicate the highest possible
 2148 priority. Priority is expected to be evenly or 'normally' distributed across this range. The
 2149 mapping of vendor-defined priority over this range is implementation-specific. [-2](#)
 2150 [indicates unknown.](#)
 2151
 2152 **jobProcessAfterDateAndTime(51), DateAndTime (SNMPv2-TC)**
 2153 OCTETS: The calendar date and time of day after which the job SHALL become a
 2154 candidate to be scheduled for processing. If the value of this attribute is in the future, the
 2155 server SHALL set the value of the job's **jmJobState** object to **pendingHeld** and add the
 2156 **jobProcessAfterSpecified** bit value to the job's **jmJobStateReasons1** object. When the
 2157 specified date and time arrives, the server SHALL remove the **jobProcessAfterSpecified**
 2158 bit value from the job's **jmJobStateReasons1** object and, if no other reasons remain,
 2159 SHALL change the job's **jmJobState** object to **pending**.
 2160
 2161 **jobHold(52), JmBooleanTC**
 2162 INTEGER: If the value is **true(4)**, a client has explicitly specified that the job is to be
 2163 held until explicitly released. Until the job is explicitly released by a client, the job
 2164 SHALL be in the **pendingHeld** state with the **jobHoldSpecified** value in the
 2165 **jmJobStateReasons1** attribute.
 2166
 2167 **jobHoldUntil(53), JmJobStringTC(SIZE(0..63))**
 2168 OCTETS: The named time period during which the job SHALL become a candidate for
 2169 processing, such as **'evening'**, **'night'**, **'weekend'**, **'second-shift'**, **'third-shift'**, etc., as
 2170 defined by the system administrator. See IPP [ipp-model] for the standard keyword
 2171 values. Until that time period arrives, the job SHALL be in the **pendingHeld** state with
 2172 the **jobHoldUntilSpecified** value in the **jmJobStateReasons1** object. The value **'no-**
 2173 **hold'** SHALL indicate explicitly that no time period has been specified; the absence of
 2174 this attribute SHALL indicate implicitly that no time period has been specified.
 2175
 2176 **outputBin(54), Integer32(0..2147483647)**
 2177 AND/OR
 2178 **JmJobStringTC(SIZE(0..63))**
 2179 INTEGER: MULTI-ROW: The output subunit index in the Printer MIB[print-mib]

2180
2181 AND/OR
2182
2183 OCTETS: MULTI-ROW: the name or number (represented as ASCII digits) of the
2184 output bin to which all or part of the job is placed in.
2185
2186 **sides(55), Integer32(-2..2)**
2187 INTEGER: MULTI-ROW: The number of sides, '1' or '2', that any document in this job
2188 requires/used.
2189
2190 **finishing(56), JmFinishingTC**
2191 INTEGER: MULTI-ROW: Type of finishing that any document in this job
2192 requires/used.
2193
2194
2195 ++++++
2196 + **Image Quality attributes (requested and consumed)**
2197 +
2198 + **For devices that can vary the image quality.**
2199 ++++++
2200
2201 **printQualityRequested(70), JmPrintQualityTC**
2202 INTEGER: MULTI-ROW: The print quality selection requested for a document in the
2203 job for printers that allow quality differentiation.
2204
2205 **printQualityUsed(71), JmPrintQualityTC**
2206 INTEGER: MULTI-ROW: The print quality selection actually used by a document in
2207 the job for printers that allow quality differentiation.
2208
2209 **printerResolutionRequested(72), JmPrinterResolutionTC**
2210 OCTETS: MULTI-ROW: The printer resolution requested for a document in the job for
2211 printers that support resolution selection.
2212
2213 **printerResolutionUsed(73), JmPrinterResolutionTC**
2214 OCTETS: MULTI-ROW: The printer resolution actually used by a document in the job
2215 for printers that support resolution selection.
2216
2217 **tonerEcomonyRequested(74), JmTonerEconomyTC**
2218 INTEGER: MULTI-ROW: The toner economy selection requested for documents in the
2219 job for printers that allow toner economy differentiation.
2220
2221 **tonerEcomonyUsed(75), JmTonerEconomyTC**
2222 INTEGER: MULTI-ROW: The toner economy selection actually used by documents in
2223 the job for printers that allow toner economy differentiation.
2224
2225 **tonerDensityRequested(76), Integer32(-2..100)**
2226 INTEGER: MULTI-ROW: The toner density requested for a document in this job for
2227 devices that can vary toner density levels. Level 1 is the lowest density and level 100 is

2228 the highest density level. Devices with a smaller range, SHALL map the 1-100 range
 2229 evenly onto the implemented range.

2231 **tonerDensityUsed(77), Integer32(-2..100)**

2232 INTEGER: MULTI-ROW: The toner density used by documents in this job for devices
 2233 that can vary toner density levels. Level 1 is the lowest density and level 100 is the
 2234 highest density level. Devices with a smaller range, SHALL map the 1-100 range evenly
 2235 onto the implemented range.

2238 ++++++
 2239 + **Job Progress attributes (requested and consumed)**

2240 +
 2241 + **Pairs of these attributes can be used by monitoring**
 2242 + **applications to show an indication of relative progress**
 2243 + **to users. See section 3.4, entitled**
 2244 + **'Monitoring Job Progress'.**

2247 **jobCopiesRequested(90), Integer32(-2..2147483647)**

2248 INTEGER: The number of copies of the entire job that are to be produced.

2250 **jobCopiesCompleted(91), Integer32(-2..2147483647)**

2251 INTEGER: The number of copies of the entire job that have been completed so far.

2253 **documentCopiesRequested(92), Integer32(-2..2147483647)**

2254 INTEGER: The total count of the number of document copies requested for the job as a
 2255 whole. If there are documents A, B, and C, and document B is specified to produce 4
 2256 copies, the number of document copies requested is 6 for the job.

2257
 2258 This attribute SHALL be used only when a job has multiple documents. The
 2259 **jobCopiesRequested** attribute SHALL be used when the job has only one document.

2261 ~~ISSUE: Would it be better/simpler to understand for **documentCopiesRequested** to be~~
 2262 ~~MULTI-VALUED, where each value is for a separate document in the multi-document~~
 2263 ~~job?~~

2265 **documentCopiesCompleted(93), Integer32(-2..2147483647)**

2266 INTEGER: The total count of the number of document copies completed so far for the
 2267 job as a whole. If there are documents A, B, and C, and document B is specified to
 2268 produce 4 copies, the number of document copies starts a 0 and runs up to 6 for the job as
 2269 the job processes.

2271 This attribute SHALL be used only when a job has multiple documents. The
 2272 **jobCopiesCompleted** attribute SHALL be used when the job has only one document.

2274 ~~ISSUE: Would it be better for **documentCopiesCompleted** to be MULTI-VALUED, where~~
 2275 ~~each value is for a separate document in the multi-document job?~~

2277 **jobKOctetsTransferred(94), Integer32(-2..2147483647)**
 2278 INTEGER: The number of K (1024) octets transferred to the server or device to which
 2279 the agent is providing access. This count is independent of the number of copies of the
 2280 job or documents that will be produced, but it is only a measure of the number of bytes
 2281 transferred to the server or device.
 2282
 2283 The agent SHALL round the actual number of octets transferred up to the next higher K.
 2284 Thus 0 octets SHALL be represented as '0', 1-1024 octets SHALL BE represented as '1',
 2285 1025-2048 SHALL be '2', etc. When the job completes, the values of the
 2286 **jmJobKOctetsPerCopyRequested** object and the **jobKOctetsTransferred** attribute
 2287 SHALL be equal.
 2288
 2289 NOTE - The **jobKOctetsTransferred** can be used with the
 2290 **jmJobKOctetsPerCopyRequested** object in order to produce a relative indication of the
 2291 progress of the job for agents that do not implement the **jmJobKOctetsProcessed** object.
 2292
 2293 **sheetCompletedeurrentCopyNumber(95), Integer32(-2..2147483647)**
 2294 INTEGER: The number of the copy being stacked for the current document. This
 2295 number starts at 0, is set to 1 when the first sheet of the first copy for each document is
 2296 being stacked and increases to the value of jobCopiesRequested.
 2297
 2298 For External Sheet Collation, this increments as each sheet is stacked, and is reset to 1
 2299 when the printer moves to stacking the next sheet number in a document. For internally
 2300 collated copies, this number increments when all sheets of the current document have
 2301 been stacked. See the **jobCollationType(97)** attribute.
 2302
 2303 **sheetCompletedeurrentDocumentNumber(96), Integer32(-2..2147483647)**
 2304 INTEGER: The ordinal number of the document in the job that is currently being
 2305 stacked. This number starts at 0, increments to 1 when the first sheet of the first
 2306 document in the job is being stacked, and increases to the value of numberOfDocuments
 2307 by the end of the job.
 2308
 2309 For uncollated or externally collated copies, this increments as each document is stacked,
 2310 and wraps back to 1 when the printer moves to printing the next document number in a
 2311 copy. For internally collated copies, this number increments when all copies of the
 2312 current document have been stacked. See the **jobCollationType(97)** attribute.
 2313
 2314 Implementations that only support one document jobs SHOULD NOT implement this
 2315 attribute.
 2316
 2317 ~~ISSUE: Instead of having the **currentDocumentNumber** attribute for the multi-~~
 2318 ~~document job implementation, how about making the **jmJobImpressionsCompleted** and~~
 2319 ~~the **currentCopyNumber** attributes multi-valued, one value for each document in the~~
 2320 ~~(multi-document) job? This makes it simpler to understand. The down side is that a~~
 2321 ~~monitoring program would have to get all the values for a multi-document job.~~
 2322 ~~Accounting programs would have to get all the values of the multi-valued attribute and~~
 2323 ~~add them up.~~
 2324

2325 **jobCollationType(97),** **JmJobCollationTypeTC**
 2326 INTEGER: The type of ~~jobsheet and document~~ collation. [See the definition of the term](#)
 2327 ['job collation' in Section 2.](#) See also Section 3.4, [entitled 'Monitoring Job Progress'](#).
 2328
 2329
 2330 ++++++
 2331 + **Impression attributes**
 2332 +
 2333 + [See the definition of the terms 'impression', 'sheet',](#)
 2334 [+ and 'page' in Section 2.](#) ~~For a print job, an impression is the marking of the~~
 2335 ~~+ entire side of a sheet. Two-sided processing involves two~~
 2336 ~~+ impressions per sheet. Two-up is the placement of two~~
 2337 ~~+ logical pages on one side of a sheet and so is still a~~
 2338 ~~+ single impression.~~
 2339 +
 2340 + See also **jmJobImpressionsPerCopyRequested** and
 2341 + **jmJobImpressionsCompleted** objects in the **jmJobTable**.
 2342 ++++++
 2343
 2344 **impressionsSpooled(110),** **Integer32(-2..2147483647)**
 2345 INTEGER: The number of impressions spooled to the server or device for the job so far.
 2346
 2347 **impressionsSentToDevice(111),** **Integer32(-2..2147483647)**
 2348 INTEGER: The number of impressions sent to the device for the job so far.
 2349
 2350 **impressionsInterpreted(112),** **Integer32(-2..2147483647)**
 2351 INTEGER: The number of impressions interpreted for the job so far.
 2352
 2353 **impressionsCompletedCurrentCopy(113),** **Integer32(-2..2147483647)**
 2354 INTEGER: The number of impressions completed by the device for the current copy of
 2355 the current document so far. For printing, the impressions completed includes
 2356 interpreting, marking, and stacking the output. For other types of job services, the
 2357 number of impressions completed includes the number of impressions processed.
 2358
 2359 This value SHALL be reset to 0 for each document in the job and for each document
 2360 copy.
 2361
 2362 **fullColorImpressionsCompleted(114),** **Integer32(-2..2147483647)**
 2363 INTEGER: The number of full color impressions completed by the device for this job so
 2364 far. For printing, the impressions completed includes interpreting, marking, and stacking
 2365 the output. For other types of job services, the number of impressions completed includes
 2366 the number of impressions processed. Full color impressions are typically defined as
 2367 those requiring 3 or more colorants, but this MAY vary by implementation.
 2368
 2369 **highlightColorImpressionsCompleted(115),** **Integer32(-2..2147483647)**
 2370 INTEGER: The number of highlight color impressions completed by the device for this
 2371 job so far. For printing, the impressions completed includes interpreting, marking, and
 2372 stacking the output. For other types of job services, the number of impressions completed
 2373 includes the number of impressions processed. Highlight color impressions are typically

2374 defined as those requiring black plus one other colorant, but this MAY vary by
2375 implementation.
2376

2377
2378 ++++++
2379 + Page attributes
2380 +
2381 + See the definition of 'impression', 'sheet', and 'page'
2382 + in Section 2. A page is a logical page. Number up can impose more than
2383 + one page on a single side of a sheet. Two-up is the
2384 + placement of two logical pages on one side of a sheet so
2385 + that each side counts as two pages.
2386 ++++++

2387
2388 **pagesRequested(130), Integer32(-2..2147483647)**
2389 INTEGER: The number of logical pages requested by the job to be processed.
2390

2391 **pagesCompleted(131), Integer32(-2..2147483647)**
2392 INTEGER: The number of logical pages completed for this job so far.
2393

2394 For implementations where multiple copies are produced by the interpreter with only a
2395 single pass over the data, the final value SHALL be equal to the value of the
2396 **pagesRequested** object. For implementations where multiple copies are produced by the
2397 interpreter by processing the data for each copy, the final value SHALL be a multiple of
2398 the value of the **pagesRequested** object.
2399

2400 NOTE - See the **impressionsCompletedCurrentCopy** and
2401 **pagesCompletedCurrentCopy** attributes for attributes that are reset on each document
2402 copy.
2403

2404 NOTE - The **pagesCompleted** object can be used with the **pagesRequested** object to
2405 provide an indication of the relative progress of the job, provided that the multiplicative
2406 factor is taken into account for some implementations of multiple copies.
2407

2408 **pagesCompletedCurrentCopy(132), Integer32(-2..2147483647)**
2409 INTEGER: The number of logical pages completed for the current copy of the document
2410 so far. This value SHALL be reset to 0 for each document in the job and for each
2411 document copy.
2412

2413
2414 ++++++
2415 + Sheet attributes
2416 +
2417 + See the definition of 'impression', 'sheet', and 'page'
2418 + in Section 2. The sheet is a single piece of a medium, whether printing
2419 + on one or both sides.
2420 ++++++
2421

2422 **sheetsRequested(150),** **Integer32(-2..2147483647)**
 2423 INTEGER: The total number of medium sheets requested to be processed for this job.
 2424
 2425 Unlike the **jmJobKOctetsPerCopyRequested** and
 2426 **jmJobImpressionsPerCopyRequested** attributes, the **sheetsRequested(150)** attribute
 2427 SHALL include the multiplicative factor contributed by the number of copies.
 2428

2429 **sheetsCompleted(151),** **Integer32(-2..2147483647)**
 2430 INTEGER: The number of medium sheets that have completed marking and stacking for
 2431 the entire job so far whether those sheets have been processed on one side or on both.
 2432

2433 **sheetsCompletedCurrentCopy(152),** **Integer32(-2..2147483647)**
 2434 INTEGER: The number of medium sheets that have completed marking and stacking for
 2435 the current copy of a document in the job so far whether those sheets have been processed
 2436 on one side or on both.
 2437
 2438 The value of this attribute SHALL be reset to **0** as each document in the job starts being
 2439 processed and for each document copy as it starts being processed.
 2440

2441
 2442 ++++++
 2443 + **Resources attributes (requested and consumed)**
 2444 +
 2445 + **Pairs of these attributes can be used by monitoring**
 2446 + **applications to show an indication of relative usage to**
 2447 + **users.**
 2448 ++++++

2449
 2450 **mediumRequested(170),** **JmMediumTypeTC**
 2451 AND/OR
 2452 **JmJobStringTC(SIZE(0..63))**
 2453 INTEGER: MULTI-ROW: The type
 2454 AND/OR
 2455 OCTETS: MULTI-ROW: the name of the medium that is required by the job.
 2456
 2457 [NOTE - The name \(**JmJobStringTC**\) values correspond to the **prtInputMediaName**](#)
 2458 [object in the Printer MIB \[print-mib\] and the values of the IPP 'media' attribute when the](#)
 2459 [attribute syntax is 'name', not 'keyword', since **mediumRequested** is in the natural](#)
 2460 [language of the job.](#)
 2461

2462 **mediumConsumed(171),** **Integer32(-2..2147483647)**
 2463 AND
 2464 **JmJobStringTC(SIZE(0..63))**
 2465 INTEGER: The number of sheets
 2466 AND
 2467 OCTETS: MULTI-ROW: MULTI-ROW: the name of the medium that has been
 2468 consumed so far whether those sheets have been processed on one side or on both.
 2469

2470 This attribute SHALL have both Integer32 and OCTET STRING (represented as
2471 JmJobStringTC) values.

2472
2473 NOTE - The name (JmJobStringTC) values correspond to the name values of the
2474 prtInputMediaName object in the Printer MIB [print-mib] and the values of the IPP
2475 'media' attribute when the attributesyntax is 'name', not 'keyword', size
2476 mediumRequested is in the natural language of the job.
2477

2478 **colorantRequested(172),** **Integer32(-2..2147483647)**
2479 **AND/OR**
2480 **JmJobStringTC(SIZE(0..63))**
2481 INTEGER: MULTI-ROW: The index (prtMarkerColorantIndex) in the Printer
2482 MIB[print-mib]
2483 AND/OR
2484 OCTETS: MULTI-ROW: the name of the colorant requested.
2485

2486 NOTE - The name (JmJobStringTC) values correspond to the name values of the
2487 prtMarkerColorantValue object in the Printer MIB. Examples are: red, blue.
2488

2489 **colorantConsumed(173),** **Integer32(-2..2147483647)**
2490 **AND/OR**
2491 **JmJobStringTC(SIZE(0..63))**
2492 INTEGER: MULTI-ROW: The index (prtMarkerColorantIndex) in the Printer
2493 MIB[print-mib]
2494 AND/OR
2495 OCTETS: MULTI-ROW: the name of the colorant consumed.
2496

2497 NOTE - The name (JmJobStringTC) values correspond to the name values of the
2498 prtMarkerColorantValue object in the Printer MIB. Examples are: red, blue
2499

2500
2501 ++++++
2502 + Time attributes (set by server or device)
2503 +
2504 + This section of attributes are ones that are set by the
2505 + server or device that accepts jobs. Two forms of time are
2506 + provided. Each form is represented in a separate attribute.
2507 + See section 3.1.2 and section 3.1.3 for the
2508 + conformance requirements for time attribute for agents and
2509 + monitoring applications, respectively. The two forms are:
2510 +
2511 + 'DateAndTime' is an 8 or 11 octet binary encoded year,
2512 + month, day, hour, minute, second, deci-second with
2513 + optional offset from UTC. See SNMPv2-TC [SMIV2-TC].
2514 +
2515 + NOTE: 'DateAndTime' is not printable characters; it is
2516 + binary.
2517 +
2518 + 'JmTimeStampTC' is the time of day measured in the number of

2519 + seconds since the system was booted.
 2520 ++++++

2521

2522 **jobSubmissionToServerTime(190),** **JmTimeStampTC**
 2523 AND/OR
 2524 **DateAndTime**

2525 INTEGER: Configuration 3 only: The time
 2526 AND/OR
 2527 OCTETS: the date and time that the job was submitted to the server (as distinguished
 2528 from the device which uses jobSubmissionTime).
 2529

2530 **jobSubmissionTime(191),** **JmTimeStampTC**
 2531 AND/OR
 2532 **DateAndTime**

2533 INTEGER: Configurations 1, 2, and 3: The time
 2534 AND/OR
 2535 OCTETS: the date and time that the job was submitted to the server or device to which
 2536 the agent is providing access.
 2537

2538

2539

2540 **jobStartedBeingHeldTime(192),** **JmTimeStampTC**
 2541 AND/OR
 2542 **DateAndTime**

2543 INTEGER: The time
 2544 AND/OR
 2545 OCTETS: the date and time that the job last entered the **pendingHeld** state. If the job
 2546 has never entered the **pendingHeld** state, then the value SHALL be '0' or the attribute
 2547 SHALL not be present in the table.
 2548

2549 **jobStartedProcessingTime(193),** **JmTimeStampTC**
 2550 AND/OR
 2551 **DateAndTime**

2552 INTEGER: The time
 2553 AND/OR
 2554 OCTETS: the date and time that the job started processing.
 2555

2556 **jobCompletionTime(194),** **JmTimeStampTC**
 2557 AND/OR
 2558 **DateAndTime**

2559 INTEGER: The time
 2560 AND/OR
 2561 OCTETS: the date and time that the job entered the **completed, canceled, or aborted**
 2562 state.
 2563

2564 **jobProcessingCPUtime(195)** **Integer32(-2..2147483647)**
 2565 **UNITS 'seconds'**
 2566 INTEGER: The amount of CPU time in seconds that the job has been in the **processing**
 2567 state. If the job enters the **processingStopped** state, that elapsed time SHALL not be

2568 included. In other words, the **jobProcessingCPUTime** value SHOULD be relatively
2569 repeatable when the same job is processed again on the same device."
2570

2571 REFERENCE

2572 "See Section 3.2 entitled 'The Attribute Mechanism' for a description of this textual-convention
2573 and its use in the **jmAttributeTable**.
2574

2575 This is a type 2 enumeration. See Section 3.7.1.2."

2576 SYNTAX INTEGER {

2577 other(1),
2578 unknown(2),
2579 jobStateReasons2(3),
2580 jobStateReasons3(4),
2581 jobStateReasons4(5),
2582 processingMessage(6),
2583 processingMessageNaturalLanguageTag(7),
2584 jobCodedCharSet(8),
2585 jobNaturalLanguageTag(9),
2586

2587 jobURI(20),
2588 jobAccountName(21),
2589 serverAssignedJobName(22),
2590 jobName(23),
2591 jobServiceTypes(24),
2592 jobSourceChannelIndex(25),
2593 jobSourcePlatformType(26),
2594 submittingServerName(27),
2595 submittingApplicationName(28),
2596 jobOriginatingHost(29),
2597 deviceNameRequested(30),
2598 queueNameRequested(31),
2599 physicalDevice(32),
2600 numberOfDocuments(33),
2601 fileName(34),
2602 documentName(35),
2603 jobComment(36),
2604 documentFormatIndex(37),
2605 documentFormat(38),
2606
2607 jobPriority(50),
2608 jobProcessAfterDateAndTime(51),
2609 jobHold(52),
2610 jobHoldUntil(53),
2611 outputBin(54),
2612 sides(55),
2613 finishing(56),
2614
2615 printQualityRequested(70),
2616 printQualityUsed(71),

```
2617     printerResolutionRequested(72),
2618     printerResolutionUsed(73),
2619     tonerEcomonyRequested(74),
2620     tonerEcomonyUsed(75),
2621     tonerDensityRequested(76),
2622     tonerDensityUsed(77),
2623
2624     jobCopiesRequested(90),
2625     jobCopiesCompleted(91),
2626     documentCopiesRequested(92),
2627     documentCopiesCompleted(93),
2628     jobKOctetsTransferred(94),
2629     sheetCompletedCurrentCopyNumber(95),
2630     sheetCompletedCurrentDocumentNumber(96),
2631     jobCollationType(97),
2632
2633     impressionsSpooled(110),
2634     impressionsSentToDevice(111),
2635     impressionsInterpreted(112),
2636     impressionsCompletedCurrentCopy(113),
2637     fullColorImpressionsCompleted(114),
2638     highlightColorImpressionsCompleted(115),
2639
2640     pagesRequested(130),
2641     pagesCompleted(131),
2642     pagesCompletedCurrentCopy(132),
2643
2644     sheetsRequested(150),
2645     sheetsCompleted(151),
2646     sheetsCompletedCurrentCopy(152),
2647
2648     mediumRequested(170),
2649     mediumConsumed(171),
2650     colorantRequested(172),
2651     colorantConsumed(173),
2652
2653     jobSubmissionToServerTime(190),
2654     jobSubmissionTime(191),
2655     jobStartedBeingHeldTime(192),
2656     jobStartedProcessingTime(193),
2657     jobCompletionTime(194),
2658     jobProcessingCPUtime(195)
2659 }
2660
2661
2662
2663
2664 JmJobServiceTypesTC ::= TEXTUAL-CONVENTION
```

2665	STATUS	current
2666	DESCRIPTION	
2667		"Specifies the type(s) of service to which the job has been submitted (print, fax, scan, etc.). The
2668		service type is represented as an enum that is bit encoded with each job service type so that
2669		more general and arbitrary services can be created, such as services with more than one
2670		destination type, or ones with only a source or only a destination. For example, a job service
2671		might scan , faxOut , and print a single job. In this case, three bits would be set in the
2672		jobServiceTypes attribute, corresponding to the hexadecimal values: 0x8 + 0x20 + 0x4 ,
2673		respectively, yielding: 0x2C .
2674		
2675		Whether this attribute is set from a job attribute supplied by the job submission client or is set
2676		by the recipient job submission server or device depends on the job submission protocol. With
2677		either implementation, the agent SHALL return a non-zero value for this attribute indicating the
2678		type of the job.
2679		
2680		One of the purposes of this attribute is to permit a requester to filter out jobs that are not of
2681		interest. For example, a printer operator MAY only be interested in jobs that include printing.
2682		That is why the attribute is in the job identification category.
2683		
2684		The following service component types are defined (in hexadecimal) and are assigned a
2685		separate bit value for use with the jobServiceTypes attribute:
2686		
2687	other	0x1
2688		The job contains some instructions that are not one of the identified types.
2689		
2690	unknown	0x2
2691		The job contains some instructions whose type is unknown to the agent.
2692		
2693	print	0x4
2694		The job contains some instructions that specify printing
2695		
2696	scan	0x8
2697		The job contains some instructions that specify scanning
2698		
2699	faxIn	0x10
2700		The job contains some instructions that specify receive fax
2701		
2702	faxOut	0x20
2703		The job contains some instructions that specify sending fax
2704		
2705	getFile	0x40
2706		The job contains some instructions that specify accessing files or documents
2707		
2708	putFile	0x80
2709		The job contains some instructions that specify storing files or documents
2710		
2711	mailList	0x100
2712		The job contains some instructions that specify distribution of documents using an
2713		electronic mail system."

2714 REFERENCE
 2715 "These bit definitions are the equivalent of a type 2 enum except that combinations of them
 2716 MAY be used together. See section 3.7.1.2."
 2717 SYNTAX INTEGER(0..2147483647) -- 31 bits, all but sign bit
 2718
 2719
 2720
 2721
 2722 **JmJobStateReasons1TC** ::= TEXTUAL-CONVENTION
 2723 STATUS current
 2724 DESCRIPTION
 2725 "The **JmJobStateReasonsN**TC ($N=1..4$) textual-conventions are used with the
 2726 **jmJobStateReasons1** object and **jobStateReasonsN** ($N=2..4$), respectively, to provide
 2727 additional information regarding the current **jmJobState** object value. These values MAY be
 2728 used with any job state or states for which the reason makes sense.
 2729
 2730 NOTE - While values cannot be added to the **jmJobState** object without impacting deployed
 2731 clients that take actions upon receiving **jmJobState** values, it is the intent that additional
 2732 **JmJobStateReasonsN**TC enums can be defined and registered without impacting such
 2733 deployed clients. In other words, the **jmJobStateReasons1** object and **jobStateReasonsN**
 2734 attributes are intended to be extensible.
 2735
 2736 NOTE - The Job Monitoring MIB contains a superset of the IPP values[ipp-model] for the IPP
 2737 'job-state-reasons' attribute, since the Job Monitoring MIB is intended to cover other job
 2738 submission protocols as well. Also some of the names of the reasons have been changed from
 2739 'printer' to 'device', since the Job Monitoring MIB is intended to cover additional types of
 2740 devices, including input devices, such as scanners.
 2741
 2742 The following standard values are defined (in hexadecimal) as *powers of two*, since multiple
 2743 values MAY be used at the same time. For ease of understanding, the
 2744 **JmJobStateReasons1TC** reasons are presented in the order in which the reasons are likely to
 2745 occur (if implemented), starting with the '**jobIncoming**' value and ending with the
 2746 '**jobCompletedWithErrors**' value.
 2747
 2748 **other** **0x1**
 2749 The job state reason is not one of the standardized or registered reasons.
 2750
 2751 **unknown** **0x2**
 2752 The job state reason is not known to the agent or is indeterminent.
 2753
 2754 **jobIncoming** **0x4**
 2755 The job has been accepted by the server or device, but the server or device is expecting
 2756 (1) additional operations from the client to finish creating the job and/or (2) is
 2757 accessing/accepting document data.
 2758

2759	submissionInterrupted	0x8
2760	The job was not completely submitted for some unforeseen reason, such as: (1) the server	
2761	has crashed before the job was closed by the client, (2) the server or the document	
2762	transfer method has crashed in some non-recoverable way before the document data was	
2763	entirely transferred to the server, (3) the client crashed or failed to close the job before the	
2764	time-out period.	
2765		
2766	jobOutgoing	0x10
2767	Configuration 2 only: The server is transmitting the job to the device.	
2768		
2769	jobHoldSpecified	0x20
2770	The value of the job's jobHold(52) attribute is TRUE. The job SHALL NOT be a	
2771	candidate for processing until this reason is removed and there are no other reasons to	
2772	hold the job.	
2773		
2774	jobHoldUntilSpecified	0x40
2775	The value of the job's jobHoldUntil(53) attribute specifies a time period that is still in the	
2776	future. The job SHALL NOT be a candidate for processing until this reason is removed	
2777	and there are no other reasons to hold the job.	
2778		
2779	jobProcessAfterSpecified	0x80
2780	The value of the job's jobProcessAfterDateAndTime(51) attribute specifies a time that is	
2781	still in the future. The job SHALL NOT be a candidate for processing until this reason is	
2782	removed and there are no other reasons to hold the job.	
2783		
2784	resourcesAreNotReady	0x100
2785	At least one of the resources needed by the job, such as media, fonts, resource objects,	
2786	etc., is not ready on any of the physical devices for which the job is a candidate. This	
2787	condition MAY be detected when the job is accepted, or subsequently while the job is	
2788	pending or processing , depending on implementation.	
2789		
2790	deviceStoppedPartly	0x200
2791	One or more, but not all, of the devices to which the job is assigned are stopped. If all of	
2792	the devices are stopped (or the only device is stopped), the deviceStopped reason	
2793	SHALL be used.	
2794		
2795	deviceStopped	0x400
2796	The device(s) to which the job is assigned is (are all) stopped.	
2797		
2798	jobInterpreting	0x800
2799	The device to which the job is assigned is interpreting the document data.	
2800		
2801	jobPrinting	0x1000
2802	The output device to which the job is assigned is marking media. This attribute is useful	
2803	for servers and output devices which spend a great deal of time processing (1) when no	
2804	marking is happening and then want to show that marking is now happening or (2) when	
2805	the job is in the process of being canceled or aborted while the job remains in the	
2806	processing state, but the marking has not yet stopped so that impression or sheet counts	
2807	are still increasing for the job.	

2808		
2809	jobCanceledByUser	0x2000
2810	The job was canceled by the owner of the job, i.e., by a user whose name is the same as	
2811	the value of the job's jmJobOwner object, or by some other authorized end-user, such as	
2812	a member of the job owner's security group.	
2813		
2814	jobCanceledByOperator	0x4000
2815	The job was canceled by the operator, i.e., by a user who has been authenticated as having	
2816	operator privileges (whether local or remote).	
2817		
2818	jobCanceledAtDevice	0x8000
2819	The job was canceled by an unidentified local user, i.e., a user at a console at the device.	
2820		
2821	abortedBySystem	0x10000
2822	The job (1) is in the process of being aborted, (2) has been aborted by the system and	
2823	placed in the ' aborted ' state, or (3) has been aborted by the system and placed in the	
2824	'pendingHeld' state, so that a user or operator can manually try the job again.	
2825		
2826	processingToStopPoint	0x20000
2827	The requester has issued an operation to cancel or interrupt the job or the server/device	
2828	has aborted the job, but the server/device is still performing some actions on the job until	
2829	a specified stop point occurs or job termination/cleanup is completed.	
2830		
2831	This reason is recommended to be used in conjunction with the processing job state to	
2832	indicate that the server/device is still performing some actions on the job while the job	
2833	remains in the processing state. After all the job's resources consumed counters have	
2834	stopped incrementing, the server/device moves the job from the processing state to the	
2835	canceled or aborted job states.	
2836		
2837	serviceOffLine	0x40000
2838	The service or document transform is off-line and accepting no jobs. All pending jobs	
2839	are put into the pendingHeld state. This situation could be true if the service's or	
2840	document transform's input is impaired or broken.	
2841		
2842	jobCompletedSuccessfully	0x80000
2843	The job completed successfully.	
2844		
2845	jobCompletedWithWarnings	0x100000
2846	The job completed with warnings.	
2847		
2848	jobCompletedWithErrors	0x200000
2849	The job completed with errors (and possibly warnings too).	
2850		
2851		
2852	The following additional job state reasons have been added to represent job states that are in	
2853	ISO DPA[iso-dpa] and other job submission protocols:	
2854		

2855 **jobPaused** **0x400000**
2856 The job has been indefinitely suspended by a client issuing an operation to suspend the
2857 job so that other jobs may proceed using the same devices. The client MAY issue an
2858 operation to resume the paused job at any time, in which case the agent SHALL remove
2859 the **jobPaused** values from the job's **jmJobStateReasons1** object and the job is
2860 eventually resumed at or near the point where the job was paused.
2861

2862 **jobInterrupted** **0x800000**
2863 The job has been interrupted while processing by a client issuing an operation that
2864 specifies another job to be run instead of the current job. The server or device will
2865 automatically resume the interrupted job when the interrupting job completes.
2866

2867 **jobRetained** **0x1000000**
2868 The job is being retained by the server or device with all of the job's document data (and
2869 submitted resources, such as fonts, logos, and forms, if any). Thus a client could issue an
2870 operation to the server or device to either (1) re-do the job (or a copy of the job) on the
2871 same server or device or (2) resubmit the job to another server or device. When a client
2872 could no longer re-do/resubmit the job, such as after the document data has been
2873 discarded, the agent SHALL remove the **jobRetained** value from the
2874 **jmJobStateReasons1** object."
2875

REFERENCE

2876 "These bit definitions are the equivalent of a type 2 enum except that combinations of bits may
2877 be used together. See section 3.7.1.2. The remaining bits are reserved for future
2878 standardization and/or registration."
2879

2880 SYNTAX **INTEGER(0..2147483647)** -- 31 bits, all but sign bit
2881

JmJobStateReasons2TC ::= TEXTUAL-CONVENTION

2887 STATUS current

DESCRIPTION

2889 "This textual-convention is used with the **jobStateReasons2** attribute to provides additional
2890 information regarding the **jmJobState** object. See the description under
2891 **JmJobStateReasons1TC** for additional information that applies to all reasons.
2892

2893 The following standard values are defined (in hexadecimal) as *powers of two*, since multiple
2894 values may be used at the same time:

2895
2896 **cascaded** **0x1**
2897 An outbound gateway has transmitted all of the job's job and document attributes and data
2898 to another spooling system.
2899

2900 **deletedByAdministrator** **0x2**
2901 The administrator has deleted the job.
2902

2903	discardTimeArrived	0x4
2904	The job has been deleted due to the fact that the time specified by the job's job-discard-time attribute has arrived.	
2905		
2906		
2907	postProcessingFailed	0x8
2908	The post-processing agent failed while trying to log accounting attributes for the job; therefore the job has been placed into the completed state with the jobRetained	
2909	jmJobStateReasons1 object value for a system-defined period of time, so the	
2910	administrator can examine it, resubmit it, etc.	
2911		
2912		
2913	jobTransforming	0x10
2914	The server/device is interpreting document data and producing another electronic	
2915	representation.	
2916		
2917	maxJobFaultCountExceeded	0x20
2918	The job has faulted several times and has exceeded the administratively defined fault	
2919	count limit.	
2920		
2921	devicesNeedAttentionTimeOut	0x40
2922	One or more document transforms that the job is using needs human intervention in order	
2923	for the job to make progress, but the human intervention did not occur within the site-	
2924	settable time-out value.	
2925		
2926	needsKeyOperatorTimeOut	0x80
2927	One or more devices or document transforms that the job is using need a specially trained	
2928	operator (who may need a key to unlock the device and gain access) in order for the job to	
2929	make progress, but the key operator intervention did not occur within the site-settable	
2930	time-out value.	
2931		
2932	jobStartWaitTimeOut	0x100
2933	The server/device has stopped the job at the beginning of processing to await human	
2934	action, such as installing a special cartridge or special non-standard media, but the job	
2935	was not resumed within the site-settable time-out value and the server/device has	
2936	transitioned the job to the pendingHeld state.	
2937		
2938	jobEndWaitTimeOut	0x200
2939	The server/device has stopped the job at the end of processing to await human action,	
2940	such as removing a special cartridge or restoring standard media, but the job was not	
2941	resumed within the site-settable time-out value and the server/device has transitioned the	
2942	job to the completed state.	
2943		
2944	jobPasswordWaitTimeOut	0x400
2945	The server/device has stopped the job at the beginning of processing to await input of the	
2946	job's password, but the password was not received within the site-settable time-out value.	
2947		
2948	deviceTimedOut	0x800
2949	A device that the job was using has not responded in a period specified by the device's	
2950	site-settable attribute.	
2951		

2952	connectingToDeviceTimeOut	0x1000
2953	The server is attempting to connect to one or more devices which may be dial-up, polled,	
2954	or queued, and so may be busy with traffic from other systems, but server was unable to	
2955	connect to the device within the site-settable time-out value.	
2956		
2957	transferring	0x2000
2958	The job is being transferred to a down stream server or downstream device.	
2959		
2960	queuedInDevice	0x4000
2961	The server/device has queued the job in a down stream server or downstream device.	
2962		
2963	jobQueued	0x8000
2964	The server/device has queued the document data.	
2965		
2966	jobCleanup	0x10000
2967	The server/device is performing cleanup activity as part of ending normal processing.	
2968		
2969	jobPasswordWait	0x20000
2970	The server/device has selected the job to be next to process, but instead of assigning	
2971	resources and starting the job processing, the server/device has transitioned the job to the	
2972	pendingHeld state to await entry of a password (and dispatched another job, if there is	
2973	one).	
2974		
2975	validating	0x40000
2976	The server/device is validating the job <i>after</i> accepting the job.	
2977		
2978	queueHeld	0x80000
2979	The operator has held the entire job set or queue.	
2980		
2981	jobProofWait	0x100000
2982	The job has produced a single proof copy and is in the pendingHeld state waiting for the	
2983	requester to issue an operation to release the job to print normally, obeying any job and	
2984	document copy attributes that were originally submitted.	
2985		
2986	heldForDiagnostics	0x200000
2987	The system is running intrusive diagnostics, so that all jobs are being held.	
2988		
2989	noSpaceOnServer	0x800000
2990	There is no room on the server to store all of the job.	
2991		
2992	pinRequired	0x1000000
2993	The System Administrator settable device policy is (1) to require PINs, and (2) to hold	
2994	jobs that do not have a pin supplied as an input parameter when the job was created.	
2995		
2996	exceededAccountLimit	0x2000000
2997	The account for which this job is drawn has exceeded its limit. This condition SHOULD	
2998	be detected before the job is scheduled so that the user does not wait until his/her job is	
2999	scheduled only to find that the account is overdrawn. This condition MAY also occur	
3000	while the job is processing either as processing begins or part way through processing.	

3001
3002 **heldForRetry** **0x4000000**
3003 The job encountered some errors that the server/device could not recover from with its
3004 normal retry procedures, but the error might not be encountered if the job is processed
3005 again in the future. Example cases are phone number busy or remote file system in-
3006 accessible. For such a situation, the server/device SHALL transition the job from the
3007 **processing** to the **pendingHeld**, rather than to the **aborted** state.
3008
3009 The following values are from the X/Open PSIS draft standard:
3010
3011 **canceledByShutdown** **0x8000000**
3012 The job was canceled because the server or device was shutdown before completing the
3013 job.
3014
3015 **deviceUnavailable** **0x10000000**
3016 This job was aborted by the system because the device is currently unable to accept jobs.
3017
3018 **wrongDevice** **0x20000000**
3019 This job was aborted by the system because the device is unable to handle this particular
3020 job; the spooler SHOULD try another device or the user should submit the job to another
3021 device.
3022
3023 **badJob** **0x40000000**
3024 This job was aborted by the system because this job has a major problem, such as an ill-
3025 formed PDL; the spooler SHOULD not even try another device. "
3026 REFERENCE
3027 "These bit definitions are the equivalent of a type 2 enum except that combinations of them
3028 may be used together. See section 3.7.1.2. See the description under **JmJobStateReasons1TC**
3029 and the **jobStateReasons2** attribute."
3030
3031 SYNTAX **INTEGER(0..2147483647)** -- 31 bits, all but sign bit
3032
3033
3034
3035
3036
3037
3038 **JmJobStateReasons3TC ::= TEXTUAL-CONVENTION**
3039 STATUS current
3040 DESCRIPTION
3041 "This textual-convention is used with the **jobStateReasons3** attribute to provides additional
3042 information regarding the **jmJobState** object. See the description under
3043 **JmJobStateReasons1TC** for additional information that applies to all reasons.
3044
3045 The following standard values are defined (in hexadecimal) as *powers of two*, since multiple
3046 values may be used at the same time:
3047
3048 **jobInterruptedByDeviceFailure** **0x1**
3049 A device or the print system software that the job was using has failed while the job was

3050 processing. The server or device is keeping the job in the **pendingHeld** state until an
3051 operator can determine what to do with the job."

3052 REFERENCE

3053 "These bit definitions are the equivalent of a type 2 enum except that combinations of them
3054 may be used together. See section 3.7.1.2. The remaining bits are reserved for future
3055 standardization and/or registration. See the description under **JmJobStateReasons1TC** and
3056 the **jobStateReasons3** attribute."

3057 SYNTAX **INTEGER(0..2147483647)** -- 31 bits, all but sign bit

3058

3059

3060

3061

3062

3063 **JmJobStateReasons4TC** ::= TEXTUAL-CONVENTION

3064 STATUS current

3065 DESCRIPTION

3066 "This textual-convention is used in the **jobStateReasons4** attribute to provides additional
3067 information regarding the **jmJobState** object. See the description under
3068 **JmJobStateReasons1TC** for additional information that applies to all reasons.

3069

3070 The following standard values are defined (in hexadecimal) as *powers of two*, since multiple
3071 values may be used at the same time:

3072

3073 none yet defined. These bits are reserved for future standardization and/or registration."

3074 REFERENCE

3075 "These bit definitions are the equivalent of a type 2 enum except that combinations of them
3076 may be used together. See section 3.7.1.2. See the description under **JmJobStateReasons1TC**
3077 and the **jobStateReasons4** attribute."

3078

3079 SYNTAX **INTEGER(0..2147483647)** -- 31 bits, all but sign bit

```

3080
3081 jobmonMIBObjects OBJECT IDENTIFIER ::= { jobmonMIB 1 }
3082
3083 -- The General Group (MANDATORY)
3084
3085 -- The jmGeneralGroup consists entirely of the jmGeneralTable.
3086
3087 jmGeneral OBJECT IDENTIFIER ::= { jobmonMIBObjects 1 }
3088
3089 jmGeneralTable OBJECT-TYPE
3090     SYNTAX      SEQUENCE OF JmGeneralEntry
3091     MAX-ACCESS  not-accessible
3092     STATUS      current
3093     DESCRIPTION
3094         "The jmGeneralTable consists of information of a general nature that are per-job-set, but are
3095         not per-job. See Section 2 entitled 'Terminology and Job Model' for the definition of a job set."
3096     REFERENCE
3097         "The MANDATORY-GROUP macro specifies that this group is MANDATORY."
3098     ::= { jmGeneral 1 }
3099
3100 jmGeneralEntry OBJECT-TYPE
3101     SYNTAX      JmGeneralEntry
3102     MAX-ACCESS  not-accessible
3103     STATUS      current
3104     DESCRIPTION
3105         "Information about a job set (queue).
3106
3107         An entry SHALL exist in this table for each job set."
3108     INDEX { jmGeneralJobSetIndex }
3109     ::= { jmGeneralTable 1 }
3110
3111 JmGeneralEntry ::= SEQUENCE {
3112     jmGeneralJobSetIndex           Integer32(1..32767),
3113     jmGeneralNumberOfActiveJobs   Integer32(0..2147483647),
3114     jmGeneralOldestActiveJobIndex Integer32(0..2147483647),
3115     jmGeneralNewestActiveJobIndex Integer32(0..2147483647),
3116     jmGeneralJobPersistence       Integer32(15..2147483647),
3117     jmGeneralAttributePersistence Integer32(15..2147483647),
3118     jmGeneralJobSetName          JmUTF8StringTC(SIZE(0..63))
3119 }
3120
3121 jmGeneralJobSetIndex OBJECT-TYPE
3122     SYNTAX      Integer32(1..32767)
3123     MAX-ACCESS  not-accessible
3124     STATUS      current
3125     DESCRIPTION
3126         "A unique value for each job set in this MIB. The jmJobTable and jmAttributeTable tables
3127         have this same index as their primary index.
3128

```


3129 The value(s) of the **jmGeneralJobSetIndex** SHALL be persistent across power cycles, so that
3130 clients that have retained **jmGeneralJobSetIndex** values will access the same job sets upon
3131 subsequent power-up.
3132

3133 An implementation that has only one job set, such as a printer with a single queue, SHALL hard
3134 code this object with the value **1**."

3135 REFERENCE
3136 "See Section 2 entitled 'Terminology and Job Model' for the definition of a job set.
3137 Corresponds to the first index in **jmJobTable** and **jmAttributeTable**."
3138 ::= { jmGeneralEntry 1 }

3139

3140 **jmGeneralNumberOfActiveJobs** OBJECT-TYPE
3141 SYNTAX Integer32(0..2147483647)
3142 MAX-ACCESS read-only
3143 STATUS current
3144 DESCRIPTION
3145 "The current number of 'active' jobs in the **jmJobIDTable**, **jmJobTable**, and
3146 **jmAttributeTable**, i.e., the total number of jobs that are in the **pending**, **processing**, or
3147 **processingStopped** states. See the **JmJobStateTC** textual-convention for the exact
3148 specification of the semantics of the job states."
3149 DEFVAL { 0 } -- no jobs
3150 ::= { jmGeneralEntry 2 }

3151

3152 **jmGeneralOldestActiveJobIndex** OBJECT-TYPE
3153 SYNTAX Integer32 (0..2147483647)
3154 MAX-ACCESS read-only
3155 STATUS current
3156 DESCRIPTION
3157 "The **jmJobIndex** of the oldest job that is still in one of the 'active' states **pending**,
3158 **processing**, or **processingStopped**). In other words, the index of the 'active' job that has been
3159 in the job tables the longest.
3160
3161 If there are no active jobs, the agent SHALL set the value of this object to **0**."
3162 REFERENCE
3163 "See Section 3.2 entitled 'The Job Tables and the Oldest Active and Newest Active Indexes' for
3164 a description of the usage of this object."
3165 DEFVAL { 0 } -- no active jobs
3166 ::= { jmGeneralEntry 3 }

3167

3168 **jmGeneralNewestActiveJobIndex** OBJECT-TYPE
3169 SYNTAX Integer32 (0..2147483647)
3170 MAX-ACCESS read-only
3171 STATUS current
3172 DESCRIPTION
3173 "The **jmJobIndex** of the newest job that is in one of the 'active' states **pending**, **processing**, or
3174 **processingStopped**). In other words, the index of the 'active' job that has been most recently
3175 added to the **job tables**.
3176

3177 When all jobs become 'inactive', i.e., enter the **pendingHeld**, **completed**, **canceled**, or **aborted**
3178 states, the agent SHALL set the value of this object to **0**."
3179 REFERENCE
3180 "See Section 3.2 entitled 'The Job Tables and the Oldest Active and Newest Active Indexes' for
3181 a description of the usage of this object."
3182 DEFVAL { 0 } -- no active jobs
3183 ::= { jmGeneralEntry 4 }

3184
3185 **jmGeneralJobPersistence** OBJECT-TYPE
3186 SYNTAX **Integer32(15..2147483647)**
3187 UNITS "seconds"
3188 MAX-ACCESS read-only
3189 STATUS current
3190 DESCRIPTION
3191 "The minimum time in seconds for this instance of the Job Set that an entry SHALL remain in
3192 the **jmJobIDTable** and **jmJobTable** after **processing** has *completed*, i.e., the minimum time in
3193 seconds starting when the job enters the **completed**, **canceled**, or **aborted** state.
3194
3195 Configuring this object is implementation-dependent.
3196
3197 This value SHALL be equal to or greater than the value of **jmGeneralAttributePersistence**.
3198 This value SHOULD be at least 60 which gives a monitoring application one minute in which
3199 to poll for job data."
3200 DEFVAL { 60 } -- one minute
3201 ::= { jmGeneralEntry 5 }

3202
3203 **jmGeneralAttributePersistence** OBJECT-TYPE
3204 SYNTAX **Integer32(15..2147483647)**
3205 UNITS "seconds"
3206 MAX-ACCESS read-only
3207 STATUS current
3208 DESCRIPTION
3209 "The minimum time in seconds for this instance of the Job Set that an entry SHALL remain in
3210 the **jmAttributeTable** after **processing** has *completed*, i.e., the time in seconds starting when
3211 the job enters the **completed**, **canceled**, or **aborted** state.
3212
3213 Configuring this object is implementation-dependent.
3214
3215 This value SHOULD be at least 60 which gives a monitoring application one minute in which
3216 to poll for job data."
3217 DEFVAL { 60 } -- one minute
3218 ::= { jmGeneralEntry 6 }

3219
3220 **jmGeneralJobSetName** OBJECT-TYPE
3221 SYNTAX **JmUTF8StringTC(SIZE(0..63))**
3222 MAX-ACCESS read-only
3223 STATUS current
3224 DESCRIPTION

3225 "The human readable name of this job set assigned by the system administrator (by means
3226 outside of this MIB). Typically, this name SHOULD be the name of the job queue. If a server
3227 or device has only a single job set, this object can be the administratively assigned name of the
3228 server or device itself. This name does not need to be unique, though each job set in a single
3229 Job Monitoring MIB SHOULD have distinct names.
3230

NOTE - If the job set corresponds to a single printer and the Printer MIB is implemented, this
3231 value SHOULD be the same as the prtGeneralPrinterName object in the draft Printer MIB. If
3232 the job set corresponds to an IPP Printer, this value SHOULD be the same as the IPP printer-
3233 name' Printerattribute.
3234

NOTE - The purpose of this object is to help the user of the job monitoring application
3236 distinguish between several job sets in implementations that support more than one job set."
3237

REFERENCE
3238 "See the OBJECT compliance macro for the minimum maximum length required for
3239 conformance."
3240

DEFVAL { ''H } -- empty string
3241
3242 ::= { jmGeneralEntry 7 }
3243
3244
3245
3246
3247

-- The Job ID Group (MANDATORY)
3248
3249

-- The **jmJobIDGroup** consists entirely of the **jmJobIDTable**.
3250
3251

jmJobID OBJECT IDENTIFIER ::= { jobmonMIBObjects 2 }
3252
3253

jmJobIDTable OBJECT-TYPE
3254
3255 SYNTAX SEQUENCE OF JmJobIDEntry
3256
3257 MAX-ACCESS not-accessible
3258
3259 STATUS current
3260
3261 DESCRIPTION
3262 "The **jmJobIDTable** provides a correspondence map (1) between the job submission ID that a
3263 client uses to refer to a job and (2) the **jmGeneralJobSetIndex** and **jmJobIndex** that the Job
3264 Monitoring MIB agent assigned to the job and that are used to access the job in all of the other
3265 tables in the MIB. If a monitoring application already knows the **jmGeneralJobSetIndex** and
3266 the **jmJobIndex** of the job it is querying, that application NEED NOT use the **jmJobIDTable**."
3267

REFERENCE
3268 "The MANDATORY-GROUP macro specifies that this group is MANDATORY."
3269
3270 ::= { jmJobID 1 }
3271

jmJobIDEntry OBJECT-TYPE
3272
3273 SYNTAX JmJobIDEntry
3274
3275 MAX-ACCESS not-accessible
3276
3277 STATUS current
3278
3279 DESCRIPTION

3273 "The map from (1) the **jmJobSubmissionID** to (2) the **jmGeneralJobSetIndex** and
 3274 **jmJobIndex**.
 3275
 3276 An entry SHALL exist in this table for each job currently known to the agent for all job sets and
 3277 job states. There MAY be more than one **jmJobIDEntry** that maps to a single job. This many
 3278 to one mapping can occur when more than one ~~network entity application program~~ along the job
 3279 submission path ~~supplies a job submission ID wishes to monitor a job~~. See Section 3.5.
 3280 However, each job SHALL appear once and in one and only one job set."
 3281 INDEX { **jmJobSubmissionID** }
 3282 ::= { jmJobIDTable 1 }
 3283
 3284 JmJobIDEntry ::= SEQUENCE {
 3285 **jmJobSubmissionID** OCTET STRING(SIZE(48)),
 3286 **jmJobIDJobSetIndex** Integer32(~~01..32767~~),
 3287 **jmJobIDJobIndex** Integer32(~~01..2147483647~~)
 3288 }
 3289
 3290 **jmJobSubmissionID** OBJECT-TYPE
 3291 SYNTAX OCTET STRING(SIZE(48))
 3292 MAX-ACCESS not-accessible
 3293 STATUS current
 3294 DESCRIPTION
 3295 "A quasi-unique 48-octet fixed-length string ID which identifies the job within a particular
 3296 client-server environment. There are multiple formats for the **jmJobSubmissionID**. Each
 3297 format SHALL be uniquely identified. See the **JmJobSubmissionIDTypeTC** textual
 3298 convention. Each format SHALL be registered using the procedures of a type 2 enum. See
 3299 section 3.7.3 entitled: 'IANA Registration of Job Submission Id Formats'.
 3300
 3301 If the requester (client or server) does not supply a job submission ID in the job submission
 3302 protocol, then the recipient (server or device) SHALL assign a job submission ID using any of
 3303 the standard formats that have been reserved for agents and adding the final 8 octets to
 3304 distinguish the ID from others submitted from the same requester.
 3305
 3306 The monitoring application, whether in the client or running separately, MAY use the job
 3307 submission ID to help identify which **jmJobIndex** was assigned by the agent, i.e., in which row
 3308 the job information is in the other tables.
 3309
 3310 NOTE - fixed-length is used so that a management application can use a shortened GetNext
 3311 varbind (in SNMPv1 and SNMPv2) in order to get the next submission ID, disregarding the
 3312 remainder of the ID in order to access jobs independent of the trailing identifier part, e.g., to get
 3313 all jobs submitted by a particular **jmJobOwner** or submitted from a particular MAC address."
 3314 REFERENCE
 3315 "See the **JmJobSubmissionIDTypeTC** textual convention.
 3316 See APPENDIX B - Support of the Job Submission ID in Job Submission Protocols."
 3317 ~~DEFVAL { ''H } empty string~~
 3318 ::= { jmJobIDEntry 1 }
 3319
 3320 **jmJobIDJobSetIndex** OBJECT-TYPE
 3321 SYNTAX Integer32(~~01..32767~~)

3322 MAX-ACCESS read-only
3323 STATUS current
3324 DESCRIPTION
3325 "This object contains the value of the **jmGeneralJobSetIndex** for the job with the
3326 **jmJobSubmissionID** value, i.e., the job set index of the job set in which the job was placed
3327 when that server or device accepted the job. This 16-bit value in combination with the
3328 **jmJobIDJobIndex** value permits the management application to access the other tables to
3329 obtain the job-specific objects for this job."
3330 REFERENCE
3331 "See **jmGeneralJobSetIndex** in the **jmGeneralTable**."
3332 DEFVAL { 0+ } -- 0 indicates no default job set index
3333 ::= { jmJobIDEntry 2 }
3334
3335 **jmJobIDJobIndex** OBJECT-TYPE
3336 SYNTAX Integer32(0+..2147483647)
3337 MAX-ACCESS read-only
3338 STATUS current
3339 DESCRIPTION
3340 "This object contains the value of the **jmJobIndex** for the job with the **jmJobSubmissionID**
3341 value, i.e., the job index for the job when the server or device accepted the job. This value, in
3342 combination with the **jmJobIDJobSetIndex** value, permits the management application to
3343 access the other tables to obtain the job-specific objects for this job."
3344 REFERENCE
3345 "See **jmJobIndex** in the **jmJobTable**."
3346 DEFVAL { 0+ } -- 0 indicates no default jmJobIndex value.
3347 ::= { jmJobIDEntry 3 }
3348
3349
3350
3351 -- The Job Group (MANDATORY)
3352
3353 -- The **jmJobGroup** consists entirely of the **jmJobTable**.
3354
3355
3356 jmJob OBJECT IDENTIFIER ::= { jobmonMIBObjects 3 }
3357
3358 jmJobTable OBJECT-TYPE
3359 SYNTAX SEQUENCE OF JmJobEntry
3360 MAX-ACCESS not-accessible
3361 STATUS current
3362 DESCRIPTION
3363 "The **jmJobTable** consists of basic job state and status information for each job in a job set that
3364 (1) monitoring applications need to be able to access in a single SNMP Get operation, (2) that
3365 have a single value per job, and (3) that SHALL always be implemented."
3366 REFERENCE
3367 "The MANDATORY-GROUP macro specifies that this group is MANDATORY."
3368 ::= { jmJob 1 }
3369
3370 jmJobEntry OBJECT-TYPE

3371 SYNTAX JmJobEntry
3372 MAX-ACCESS not-accessible
3373 STATUS current
3374 DESCRIPTION
3375 "Basic per-job state and status information.
3376
3377 An entry SHALL exist in this table for each job, no matter what the state of the job is. Each job
3378 SHALL appear in one and only one job set."
3379 REFERENCE
3380 "See Section 3.2 entitled 'The Job Tables'."
3381 INDEX { **jmGeneralJobSetIndex**, **jmJobIndex** }
3382 ::= { jmJobTable 1 }
3383
3384 JmJobEntry ::= SEQUENCE {
3385 **jmJobIndex** Integer32(1..2147483647),
3386 **jmJobState** JmJobStateTC,
3387 **jmJobStateReasons1** JmJobStateReasons1TC,
3388 **jmNumberOfInterveningJobs** Integer32(-2..2147483647),
3389 **jmJobKOctetsPerCopyRequested** Integer32(-2..2147483647),
3390 **jmJobKOctetsProcessed** Integer32(-2..2147483647),
3391 **jmJobImpressionsPerCopyRequested** Integer32(-2..2147483647),
3392 **jmJobImpressionsCompleted** Integer32(-2..2147483647),
3393 **jmJobOwner** JmJobStringTC(SIZE(0..63))
3394 }
3395
3396 **jmJobIndex** OBJECT-TYPE
3397 SYNTAX Integer32(1..2147483647)
3398 MAX-ACCESS not-accessible
3399 STATUS current
3400 DESCRIPTION
3401 "The sequential, monotonically increasing identifier index for the job generated by the server or
3402 device when that server or device accepted the job. This index value permits the management
3403 application to access the other tables to obtain the job-specific row entries."
3404 REFERENCE
3405 "See Section 3.2 entitled 'The Job Tables and the Oldest Active and Newest Active Indexes'.
3406 See Section 3.5 entitled 'Job Identification'.
3407 See also **jmGeneralNewestActiveJobIndex** for the largest value of **jmJobIndex**.
3408 See **JmJobSubmissionIDTypeTC** for a limit on the size of this index if the agent represents it
3409 as an 8-digit decimal number."
3410 ::= { jmJobEntry 1 }
3411
3412 **jmJobState** OBJECT-TYPE
3413 SYNTAX JmJobStateTC
3414 MAX-ACCESS read-only
3415 STATUS current
3416 DESCRIPTION
3417 "The current state of the job (**pending**, **processing**, **completed**, etc.). Agents SHALL
3418 implement only those states which are appropriate for the particular implementation. However,
3419 management applications SHALL be prepared to receive all the standard job states.

3420
 3421 The final value for this object SHALL be one of: **completed**, **canceled**, or **aborted**. The
 3422 minimum length of time that the agent SHALL maintain MIB data for a job in the **completed**,
 3423 **canceled**, or **aborted** state before removing the job data from the **jmJobIDTable** and
 3424 **jmJobTable** is specified by the value of the **jmGeneralJobPersistence** object."
 3425 DEFVAL { unknown } -- default is unknown
 3426 ::= { jmJobEntry 2 }
 3427

3428 **jmJobStateReasons1** OBJECT-TYPE
 3429 SYNTAX **JmJobStateReasons1TC**
 3430 MAX-ACCESS read-only
 3431 STATUS current
 3432 DESCRIPTION
 3433 "Additional information about the job's current state, i.e., information that augments the value
 3434 of the job's **jmJobState** object.
 3435
 3436 Implementation of any reason values is OPTIONAL, but an agent SHOULD return any reason
 3437 information available. These values MAY be used with any job state or states for which the
 3438 reason makes sense. Since the Job State Reasons will be more dynamic than the Job State, it is
 3439 recommended that a job monitoring application read this object every time **jmJobState** is read.
 3440 When the agent cannot provide a reason for the current state of the job, the value of the
 3441 **jmJobStateReasons1** object and **jobStateReasonsN** attributes SHALL be 0."
 3442 REFERENCE
 3443 "The **jobStateReasonsN** ($N=2..4$) attributes provide further additional information about the
 3444 job's current state."
 3445 DEFVAL { 0 } -- no reasons
 3446 ::= { jmJobEntry 3 }
 3447

3448 **jmNumberOfInterveningJobs** OBJECT-TYPE
 3449 SYNTAX **Integer32(-2..2147483647)**
 3450 MAX-ACCESS read-only
 3451 STATUS current
 3452 DESCRIPTION
 3453 "The number of jobs that are expected to complete processing *before* this job has completed
 3454 processing according to the implementation's queuing algorithm, if no other jobs were to be
 3455 submitted. In other words, this value is the job's queue position. The agent SHALL return a
 3456 value of 0 for this attribute when the job is the next job to complete processing (or has
 3457 completed processing)."
 3458 DEFVAL { 0 } -- default is no intervening jobs.
 3459 ::= { jmJobEntry 4 }
 3460

3461 **jmJobKOctetsPerCopyRequested** OBJECT-TYPE
 3462 SYNTAX **Integer32(-2..2147483647)**
 3463 MAX-ACCESS read-only
 3464 STATUS current
 3465 DESCRIPTION
 3466 "The total size in K (1024) octets of the document(s) being requested to be processed in the job.
 3467 The agent SHALL round the actual number of octets up to the next highest K. Thus 0 octets

3468 SHALL be represented as **0'**, 1-1024 octets SHALL be represented as **1'**, 1025-2048 SHALL
 3469 be represented as **2'**, etc.

3470
 3471 In computing this value, the server/device SHALL *not* include the multiplicative factors
 3472 contributed by (1) the number of document copies, and (2) the number of job copies,
 3473 independent of whether the device can process multiple copies of the job or document without
 3474 making multiple passes over the job or document data and independent of whether the output is
 3475 collated or not. Thus the server/device computation is independent of the implementation and
 3476 reflects the size of the document(s) independent of the number of copies."

3477 DEFVAL { -2 } -- the default is unknown(-2)

3478 ::= { jmJobEntry 5 }

3479

3480 **jmJobKOctetsProcessed** OBJECT-TYPE

3481 SYNTAX **Integer32(-2..2147483647)**

3482 MAX-ACCESS read-only

3483 STATUS current

3484 DESCRIPTION

3485 "The total number of octets processed by the server or device measured in units of K (1024)
 3486 octets so far. The agent SHALL round the actual number of octets processed up to the next
 3487 higher K. Thus 0 octets SHALL be represented as **0'**, 1-1024 octets SHALL be represented as
 3488 **1'**, 1025-2048 octets SHALL be **2'**, etc. For printing devices, this value is the number
 3489 interpreted by the page description language interpreter rather than what has been marked on
 3490 media.

3491

3492 For implementations where multiple copies are produced by the interpreter with only a single
 3493 pass over the data, the final value SHALL be equal to the value of the

3494 **jmJobKOctetsPerCopyRequested** object. For implementations where multiple copies are
 3495 produced by the interpreter by processing the data for each copy, the final value SHALL be a
 3496 multiple of the value of the **jmJobKOctetsPerCopyRequested** object.

3497

3498 NOTE - See the **impressionsCompletedCurrentCopy** and **pagesCompletedCurrentCopy**
 3499 attributes for attributes that are reset on each document copy.

3500

3501 NOTE - The **jmJobKOctetsProcessed** object can be used with the
 3502 **jmJobKOctetsPerCopyRequested** object to provide an indication of the relative progress of
 3503 the job, provided that the multiplicative factor is taken into account for some implementations
 3504 of multiple copies."

3505 DEFVAL { 0 } -- default is no octets processed.

3506 ::= { jmJobEntry 6 }

3507

3508 **jmJobImpressionsPerCopyRequested** OBJECT-TYPE

3509 SYNTAX **Integer32(-2..2147483647)**

3510 MAX-ACCESS read-only

3511 STATUS current

3512 DESCRIPTION

3513 "The total size in number of impressions of the document(s) being requested by this job to
 3514 produce.

3515

3516 In computing this value, the server/device SHALL *not* include the multiplicative factors
 3517 contributed by (1) the number of document copies, and (2) the number of job copies,
 3518 independent of whether the device can process multiple copies of the job or document without
 3519 making multiple passes over the job or document data and independent of whether the output is
 3520 collated or not. Thus the server/device computation is independent of the implementation and
 3521 reflects the size of the document(s) independent of the number of copies."

REFERENCE

"[See the definition of the term "impression" in Section 2.](#)"

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

::= { jmJobEntry 7 }

jmJobImpressionsCompleted OBJECT-TYPE

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

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"The total number of impressions completed for this job so far. For printing devices, the impressions completed includes interpreting, marking, and stacking the output. For other types of job services, the number of impressions completed includes the number of impressions processed.

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

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

REFERENCE

"[See the definition of the term "impression" in Section 2 and the counting example in Section 3.4 entitled "Monitoring Job Progress."](#)

DEFVAL { 0 } -- default is no octets

::= { jmJobEntry 8 }

jmJobOwner OBJECT-TYPE

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

MAX-ACCESS read-only

STATUS current

DESCRIPTION

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

This value SHOULD be the most *authenticated* name of the user submitting the job.

NOTE - This attribute corresponds to the IPP 'job-originating-user-name' job description attribute, which MAY be derived from the 'requesting-user-name' operation attribute if a more authenticated name is not available."

REFERENCE

```

3565         "See the OBJECT compliance macro for the minimum maximum length required for
3566         conformance."
3567     DEFVAL    { ''H }    -- empty string
3568     ::= { jmJobEntry 9 }
3569
3570
3571
3572
3573 -- The Attribute Group (MANDATORY)
3574
3575 -- The jmAttributeGroup consists entirely of the jmAttributeTable.
3576 --
3577 -- Implementation of the two objects in this group is MANDATORY.
3578 -- See Section 3.1 entitled 'Conformance Considerations'.
3579 -- An agent SHALL implement any attribute if (1) the server or device
3580 -- supports the functionality represented by the attribute and (2) the
3581 -- information is available to the agent.
3582
3583 jmAttribute OBJECT IDENTIFIER ::= { jobmonMIBObjects 4 }
3584
3585 jmAttributeTable OBJECT-TYPE
3586     SYNTAX      SEQUENCE OF JmAttributeEntry
3587     MAX-ACCESS  not-accessible
3588     STATUS      current
3589     DESCRIPTION
3590         "The jmAttributeTable SHALL contain attributes of the job and document(s) for each job in a
3591         job set. Instead of allocating distinct objects for each attribute, each attribute is represented as a
3592         separate row in the jmAttributeTable."
3593     REFERENCE
3594         "The MANDATORY-GROUP macro specifies that this group is MANDATORY. An agent
3595         SHALL implement any attribute if (1) the server or device supports the functionality
3596         represented by the attribute and (2) the information is available to the agent. "
3597     ::= { jmAttribute 1 }
3598
3599 jmAttributeEntry OBJECT-TYPE
3600     SYNTAX      JmAttributeEntry
3601     MAX-ACCESS  not-accessible
3602     STATUS      current
3603     DESCRIPTION
3604         "Attributes representing information about the job and document(s) or resources required and/or
3605         consumed.
3606
3607         Each entry in the jmAttributeTable is a per-job entry with an extra index for each type of
3608         attribute (jmAttributeTypeIndex) that a job can have and an additional index
3609         (jmAttributeInstanceIndex) for those attributes that can have multiple instances per job. The
3610         jmAttributeTypeIndex object SHALL contain an enum type that indicates the type of attribute
3611         (see the JmAttributeTypeTC textual-convention). The value of the attribute SHALL be
3612         represented in either the jmAttributeValueAsInteger or jmAttributeValueAsOctets objects,
3613         and/or both, as specified in the JmAttributeTypeTC textual-convention.

```

3614
3615 The agent SHALL create rows in the **jmAttributeTable** as the server or device is able to
3616 discover the attributes either from the job submission protocol itself or from the document
3617 PDL. As the documents are interpreted, the interpreter MAY discover additional attributes and
3618 so the agent adds additional rows to this table. As the attributes that represent resources are
3619 actually consumed, the usage counter contained in the **jmAttributeValueAsInteger** object is
3620 incremented according to the units indicated in the description of the **JmAttributeTypeTC**
3621 enum.
3622
3623 The agent SHALL maintain each row in the **jmJobTable** for at least the minimum time after a
3624 job completes as specified by the **jmGeneralAttributePersistence** object.
3625
3626 Zero or more entries SHALL exist in this table for each job in a job set."
3627 REFERENCE
3628 "See Section 3.3 entitled 'The Attribute Mechanism' for a description of the
3629 **jmAttributeTable**."
3630 INDEX { **jmGeneralJobSetIndex**, **jmJobIndex**, **jmAttributeTypeIndex**,
3631 **jmAttributeInstanceIndex** }
3632 ::= { jmAttributeTable 1 }
3633
3634 JmAttributeEntry ::= SEQUENCE {
3635 **jmAttributeTypeIndex** **JmAttributeTypeTC**,
3636 **jmAttributeInstanceIndex** **Integer32(1..32767)**,
3637 **jmAttributeValueAsInteger** **Integer32(-2..2147483647)**,
3638 **jmAttributeValueAsOctets** **OCTET STRING(SIZE(0..63))**
3639 }
3640
3641 **jmAttributeTypeIndex** OBJECT-TYPE
3642 SYNTAX **JmAttributeTypeTC**
3643 MAX-ACCESS not-accessible
3644 STATUS current
3645 DESCRIPTION
3646 "The type of attribute that this row entry represents.
3647
3648 The type MAY identify information about the job or document(s) or MAY identify a resource
3649 required to process the job before the job start processing and/or consumed by the job as the job
3650 is processed.
3651
3652 Examples of job attributes (i.e., apply to the job as a whole) that have only one instance per job
3653 include: **jobCopiesRequested(90)**, **documentCopiesRequested(92)**,
3654 **jobCopiesCompleted(91)**, **documentCopiesCompleted(93)**, while examples of job attributes
3655 that may have more than one instance per job include: **documentFormatIndex(37)**, and
3656 **documentFormat(38)**.
3657
3658 Examples of document attributes (one instance per document) include: **fileName(34)**, and
3659 **documentName(35)**.
3660
3661 Examples of required and consumed resource attributes include: **pagesRequested(130)**,
3662 **mediumRequested(170)**, **pagesCompleted(131)**, and **mediumConsumed(171)**, respectively."

3663 ::= { jmAttributeEntry 1 }

3664

3665 **jmAttributeInstanceIndex** OBJECT-TYPE

3666 SYNTAX **Integer32(1..32767)**

3667 MAX-ACCESS not-accessible

3668 STATUS current

3669 DESCRIPTION

3670 "A running 16-bit index of the attributes of the same type for each job. For those attributes with
 3671 only a single instance per job, this index value SHALL be 1. For those attributes that are a
 3672 single value per document, the index value SHALL be the document number, starting with 1 for
 3673 the first document in the job. Jobs with only a single document SHALL use the index value of
 3674 1. For those attributes that can have multiple values per job or per document, such as
 3675 **documentFormatIndex(37)** or **documentFormat(38)**, the index SHALL be a running index
 3676 for the job as a whole, starting at 1."

3677 ::= { jmAttributeEntry 2 }

3678

3679 **jmAttributeValueAsInteger** OBJECT-TYPE

3680 SYNTAX **Integer32(-2..2147483647)**

3681 MAX-ACCESS read-only

3682 STATUS current

3683 DESCRIPTION

3684 "The integer value of the attribute. The value of the attribute SHALL be represented as an
 3685 integer if the enum description in the **JmAttributeTypeTC** textual-convention definition has
 3686 the tag: 'INTEGER:'.

3687

3688 Depending on the enum definition, this object value MAY be an integer, a counter, an index, or
 3689 an enum, depending on the **jmAttributeTypeIndex** value. The units of this value are specified
 3690 in the enum description.

3691

3692 For those attributes that are accumulating job consumption as the job is processed as specified
 3693 in the **JmAttributeTypeTC** textual-convention, SHALL contain the final value after the job
 3694 completes processing, i.e., this value SHALL indicate the total usage of this resource made by
 3695 the job.

3696

3697 A monitoring application is able to copy this value to a suitable longer term storage for later
 3698 processing as part of an accounting system.

3699

3700 Since the agent MAY add attributes representing resources to this table while the job is waiting
 3701 to be processed or being processed, which can be a long time before any of the resources are
 3702 actually used, the agent SHALL set the value of the **jmAttributeValueAsInteger** object to 0
 3703 for resources that the job has not yet consumed.

3704

3705 Attributes for which the concept of an integer value is meaningless, such as **fileName(34)**,
 3706 **jobName**, and **processingMessage**, do *not* have the 'INTEGER:' tag in the
 3707 **JmAttributeTypeTC** definition and so an agent SHALL always return a value of '-1' to
 3708 indicate 'other' for the value of the **jmAttributeValueAsInteger** object for these attributes.

3709

3710 For attributes which do have the 'INTEGER:' tag in the **JmAttributeTypeTC** definition, if the
 3711 integer value is not (yet) known, the agent either (1) SHALL not materialize the row in the

3712 **jmAttributeTable** until the value is known or (2) SHALL return a '-2' to represent an
3713 'unknown' counting integer value, a 0' to represent an 'unknown' index value, and a 2' to
3714 represent an 'unknown(2)' enum value."
3715 DEFVAL { -2 } -- default value is unknown(-2)
3716 ::= { jmAttributeEntry 3 }
3717
3718 **jmAttributeValueAsOctets** OBJECT-TYPE
3719 SYNTAX OCTET STRING(SIZE(0..63))
3720 MAX-ACCESS read-only
3721 STATUS current
3722 DESCRIPTION
3723 "The octet string value of the attribute. The value of the attribute SHALL be represented as an
3724 OCTET STRING if the enum description in the **JmAttributeTypeTC** textual-convention
3725 definition has the tag: 'OCTETS:'.
3726
3727 Depending on the enum definition, this object value MAY be a coded character set string (text),
3728 such as '**JmUTF8StringTC**', or a binary octet string, such as '**DateAndTime**'.
3729
3730 Attributes for which the concept of an octet string value is meaningless, such as
3731 **pagesCompleted**, do *not* have the tag 'OCTETS:' in the **JmAttributeTypeTC** definition and so
3732 the agent SHALL always return a zero length string for the value of the
3733 **jmAttributeValueAsOctets** object.
3734
3735 For attributes which do have the 'OCTETS:' tag in the **JmAttributeTypeTC** definition, if the
3736 OCTET STRING value is not (yet) known, the agent either SHALL not materialize the row in
3737 the **jmAttributeTable** until the value is known or SHALL return a zero-length string."
3738 DEFVAL { ''H } -- empty string
3739 ::= { jmAttributeEntry 4 }
3740


```
3741 -- Notifications and Trapping
3742 -- Reserved for the future
3743
3744 jobmonMIBNotifications OBJECT IDENTIFIER ::= { jobmonMIB 2 }
3745
3746
3747
3748 -- Conformance Information
3749
3750 jmMIBConformance OBJECT IDENTIFIER ::= { jobmonMIB 3 }
3751
3752 -- compliance statements
3753 jmMIBCompliance MODULE-COMPLIANCE
3754     STATUS current
3755     DESCRIPTION
3756         "The compliance statement for agents that implement the
3757         job monitoring MIB."
3758     MODULE -- this module
3759     MANDATORY-GROUPS {
3760         jmGeneralGroup, jmJobIDGroup, jmJobGroup, jmAttributeGroup }
3761
3762     OBJECT jmGeneralJobSetName
3763     SYNTAX JmUTF8StringTC (SIZE(0..8))
3764     DESCRIPTION
3765         "Only 8 octets maximum string length NEED be supported by the agent."
3766
3767     OBJECT jmJobOwner
3768     SYNTAX JmJobStringTC (SIZE(0..16))
3769     DESCRIPTION
3770         "Only 16 octets maximum string length NEED be supported by the agent."
3771
3772 -- There are no CONDITIONALLY MANDATORY or OPTIONAL groups.
3773
3774     ::= { jmMIBConformance 1 }
3775
3776 jmMIBGroups OBJECT IDENTIFIER ::= { jmMIBConformance 2 }
3777
3778 jmGeneralGroup OBJECT-GROUP
3779     OBJECTS {
3780         jmGeneralNumberOfActiveJobs, jmGeneralOldestActiveJobIndex,
3781         jmGeneralNewestActiveJobIndex, jmGeneralJobPersistence,
3782         jmGeneralAttributePersistence, jmGeneralJobSetName }
3783     STATUS current
3784     DESCRIPTION
3785         "The general group."
3786     ::= { jmMIBGroups 1 }
3787
3788 jmJobIDGroup OBJECT-GROUP
3789     OBJECTS {
```



```
3790         jmJobIDJobSetIndex, jmJobIDJobIndex }
3791     STATUS current
3792     DESCRIPTION
3793         "The job ID group."
3794     ::= { jmMIBGroups 2 }
3795
3796 jmJobGroup OBJECT-GROUP
3797     OBJECTS {
3798         jmJobState, jmJobStateReasons1, jmNumberOfInterveningJobs,
3799         jmJobKOctetsPerCopyRequested, jmJobKOctetsProcessed,
3800         jmJobImpressionsPerCopyRequested, jmJobImpressionsCompleted, jmJobOwner }
3801     STATUS current
3802     DESCRIPTION
3803         "The job group."
3804     ::= { jmMIBGroups 3 }
3805
3806 jmAttributeGroup OBJECT-GROUP
3807     OBJECTS {
3808         jmAttributeValueAsInteger, jmAttributeValueAsOctets }
3809     STATUS current
3810     DESCRIPTION
3811         "The attribute group."
3812     ::= { jmMIBGroups 4 }
3813
3814
3815     END
```

3816 5. Appendix A - Implementing the Job Life Cycle

3817 The job object has well-defined states and client operations that affect the transition between the
3818 job states. Internal server and device actions also affect the transitions of the job between the job
3819 states. These states and transitions are referred to as the job's *life cycle*.

3820 Not all implementations of job submission protocols have all of the states of the job model
3821 specified here. The job model specified here is intended to be a superset of most
3822 implementations. It is the purpose of the agent to map the particular implementation's job life
3823 cycle onto the one specified here. The agent MAY omit any states not implemented. Only the
3824 **processing** and **completed** states are required to be implemented by an agent. However, a
3825 conforming management application SHALL be prepared to accept any of the states in the job
3826 life cycle specified here, so that the management application can interoperate with any
3827 conforming agent.

3828 The job states are intended to be user visible. The agent SHALL make these states visible in the
3829 MIB, but only for the subset of job states that the implementation has. Some implementations
3830 MAY need to have sub-states of these user-visible states. The **jmJobStateReasons1** object and
3831 the **jobStateReasonsN** ($N=2..4$) attributes can be used to represent the sub-states of the jobs.

3832 Job states are intended to last a user-visible length of time in most implementations. However,
3833 some jobs may pass through some states in zero time in some situations and/or in some
3834 implementations.

3835 The job model does not specify how accounting and auditing is implemented, except to assume
3836 that accounting and auditing logs are separate from the job life cycle and last longer than job
3837 entries in the MIB. Jobs in the **completed**, **aborted**, or **canceled** states are not logs, since jobs in
3838 these states are accessible via SNMP protocol operations and SHALL be removed from the Job
3839 Monitoring MIB tables after a site-settable or implementation-defined period of time. An
3840 accounting application MAY copy accounting information incrementally to an accounting log as
3841 a job processes, or MAY be copied while the job is in the **canceled**, **aborted**, or **completed**
3842 states, depending on implementation. The same is true for auditing logs.

3843 **The jmJobState object specifies the standard job states. The normal job state transitions**
3844 **are shown in the state transition diagram presented in Table 1.**

3845 6. APPENDIX B - Support of the Job Submission ID in Job Submission 3846 Protocols

3847 This appendix lists the job submission protocols that support the concept of a job
3848 submission ID and indicates the attribute used in that job submission protocol.

3849 **6.1 Hewlett-Packard's Printer Job Language (PJL)**

3850 Hewlett-Packard's Printer Job Language provides job-level printer control and printer
3851 status information to applications. The PJL JOB command is used at the beginning of a
3852 print job and can include options applying only to that job. A PJL JOB command option
3853 has been defined to facilitate passing the **JobSubmissionID** with the print job, as
3854 required by the Job Monitoring MIB. The option is of the form:

```
3855  
3856     SUBMISSIONID = "id string"  
3857
```

3858 Where the "id string" is a string and SHALL be enclosed in double quotes. The format is
3859 as described for the **jmJobSubmissionID** object.

3860 The entire PJL JOB command with the optional parameter would be of the form:

```
3861  
3862     @PJL JOB SUBMISSIONID = "id string"  
3863
```

3864 See "Printer Job Language Technical Reference Manual", part number 5021-0328, from
3865 Hewlett-Packard for complete information on the PJL JOB command and the Printer Job
3866 Language.

3867 NOTE - Some PJL implementations wrap a banner page as a PJL job around a job
3868 submitted by a client. If this results in multiple job submission IDs~~In this case, there will~~
3869 ~~be two job submission ids. The outer one being the one with the banner page and the~~
3870 ~~inner one being the original user's job. The~~ agent SHALL create multiple
3871 jmJobIDEntry rows in the jmJobIDTable that each point to the same job entry in the
3872 job tables~~use the last received job submission ID for the jmJobSubmissionID index, so~~
3873 ~~that the original user's job submission ID will be used, not the banner page job ID. See~~
3874 the specification of the jmJobIDEntry.

3875 **6.2 ISO DPA**

3876 The ISO 10175 Document Printing Application (DPA) protocol specifies the "**job-client-**
3877 **id**" attribute that allows the client to supply a text string ID for each job.

3878 **7. References**

3879 [char-set policy] Harald Avelstrand, "IETF Policy on Character Sets and Language",
3880 June 1997. Latest draft: <draft-avelstrand-charset-policy-00.txt>

3881 [GB2312] GB 2312-1980, "Chinese People's Republic of China (PRC) mixed one byte
3882 and two byte coded character set"

3883 [hr-mib] P. Grillo, S. Waldbusser, "Host Resources MIB", RFC 1514, September 1993

- 3884 [iana] J. Reynolds, and J. Postel, "Assigned Numbers", STD 2, RFC 1700, ISI, October
3885 1994.
- 3886 [IANA-charsets] Coded Character Sets registered by IANA and assigned an enum value
3887 for use in the **CodedCharSet** textual convention imported from the Printer MIB. See
3888 <ftp://ftp.isi.edu/in-notes/iana/assignments/character-sets>
- 3889 [iana-media-types] IANA Registration of MIME media types (MIME content
3890 types/subtypes). See <ftp://ftp.isi.edu/in-notes/iana/assignments/>
- 3891 [ISO-639] ISO 639:1988 (E/F) - Code for Representation of names of languages - The
3892 International Organization for Standardization, 1st edition, 1988.
- 3893 [ISO 646] ISO/IEC 646:1991, "Information technology -- ISO 7-bit coded character set
3894 for information interchange", JTC1/SC2.
- 3895 [ISO 8859] ISO/IEC 8859-1:1987, "Information technology -- 8-bit single byte coded
3896 graphic character sets - Part 1: Latin alphabet No. 1, JTC1/SC2."
- 3897 [ISO 2022] ISO/IEC 2022:1994 - "Information technology -- Character code structure
3898 and extension techniques", JTC1/SC2.
- 3899 [ISO-3166] ISO 3166:1988 (E/F) - Codes for representation of names of countries - The
3900 International Organization for Standardization, 3rd edition, 1988-08-15."
- 3901 [ISO-10646] ISO/IEC 10646-1:1993, "Information technology -- Universal Multiple-
3902 Octet Coded Character Set (UCS) - Part 1: Architecture and Basic Multilingual Plane,
3903 JTC1/SC2.
- 3904 [iso-dpa] ISO/IEC 10175 Document Printing Application (DPA). See
3905 <ftp://ftp.pwg.org/pub/pwg/dpa/>
- 3906 [ipp-model] Internet Printing Protocol (IPP), work in progress on the IETF standards
3907 track. See **draft-ietf-ipp-model-07.txt**. See also <http://www.pwg.org/ipp/index.html>
- 3908 [JIS X0208] JIS X0208-1990, "Japanese two byte coded character set."
- 3909 [mib-II] MIB-II, RFC 1213.
- 3910 [print-mib] The Printer MIB - RFC 1759, proposed IETF standard. Also an Internet-
3911 Draft on the standards track as a draft standard: **draft-ietf-printmib-mib-info-02.txt**
- 3912 [req-words] S. Bradner, "Keywords for use in RFCs to Indicate Requirement Levels",
3913 RFC 2119, March 1997.
- 3914 [rfc 1738] Berners-Lee, T., Masinter, L., McCahill, M., "Uniform Resource Locators
3915 (URL)", RFC 1738, December 1994.
- 3916 [RFC-1766] Avelstrand H., "Tags for the Identification of Languages", RFC 1766, March
3917 1995.

- 3918 [rfc 2130] C. Weider, C. Preston, K. Simonsen, H. Alvestrand, R. Atkinson, M. Crispin,
3919 and P. Svanberg, "The Report of the IAB Character Set Workshop held 29 Feb-1 March,
3920 1997", April 1997, RFC 2130.
- 3921 [\[SMIv2-SMI\] J. Case, et al. "Structure of Management Information for Version 2 of the](#)
3922 [Simple Network Management Protocol \(SNMPv2\)", RFC 1902, January 1996.](#)
- 3923 [SMIv2-TC] J. Case, et al. "Textual Conventions for Version 2 of the Simple Network
3924 Management Protocol (SNMPv2)", RFC 1903, January 1996.
- 3925 [tipsi] IEEE 1284.1, Transport-independent Printer System Interface (TIPSI).
- 3926 [URI-spec] Berners-Lee, T., Masinter, L., McCahill, M. , "Uniform Resource Locators
3927 (URL)", RFC 1738, December, 1994.
- 3928 [US-ASCII] Coded Character Set - 7-bit American Standard Code for Information
3929 Interchange, ANSI X3.4-1986.
- 3930 [UTF-8] F. Yergeau, "UTF-8, a transformation format of Unicode and ISO 10646", RFC
3931 2044, October 1996.

3932 8. Author's Addresses

- 3933 Ron Bergman
3934 Dataproducts Corp.
3935 1757 Tapo Canyon Road
3936 Simi Valley, CA 93063-3394
3937
3938 Phone: 805-578-4421
3939 Fax: 805-578-4001
3940 Email: rbergman@dpc.com
3941
3942
- 3943 Tom Hastings
3944 Xerox Corporation, ESAE-231
3945 701 S. Aviation Blvd.
3946 El Segundo, CA 90245
3947
3948 Phone: 310-333-6413
3949 Fax: 310-333-5514
3950 EMail: hastings@cp10.es.xerox.com
3951
3952
- 3953 Scott A. Isaacson
3954 Novell, Inc.

3955 122 E 1700 S
3956 Provo, UT 84606
3957
3958 Phone: 801-861-7366
3959 Fax: 801-861-4025
3960 EMail: scott_isaacson@novell.com

3961
3962

3963 Harry Lewis
3964 IBM Corporation
3965 6300 Diagonal Hwy
3966 Boulder, CO 80301

3967
3968 Phone: (303) 924-5337
3969 Fax:
3970 Email: harryl@us.ibm.com

3971
3972

3973 Send comments to the printmib WG using the Job Monitoring Project (JMP)
3974 Mailing List: jmp@pwg.org

3975

3976 To learn how to subscribe, send email to: jmp-request@pwg.org

3977

3978 For further information, access the PWG web page under "JMP":
3979 <http://www.pwg.org/>

3980

3981 Other Participants:

3982 Chuck Adams - Tektronix
3983 Jeff Barnett - IBM
3984 Keith Carter, IBM Corporation
3985 Jeff Copeland - QMS
3986 Andy Davidson - Tektronix
3987 Roger deBry - IBM
3988 Mabry Dozier - QMS
3989 Lee Ferrel - Canon
3990 Steve Gebert - IBM
3991 Robert Herriot - Sun Microsystems Inc.
3992 Shige Kanemitsu - Kyocera
3993 David Kellerman - Northlake Software
3994 Rick Landau - Digital

3995 Harry Lewis - IBM
3996 Pete Loya - HP
3997 Ray Lutz - Cognisys
3998 Jay Martin - Underscore
3999 Mike MacKay, Novell, Inc.
4000 Stan McConnell - Xerox
4001 Carl-Uno Manros, Xerox, Corp.
4002 Pat Nogay - IBM
4003 Bob Pentecost - HP
4004 Rob Rhoads - Intel
4005 David Roach - Unisys
4006 [Stuart Rowley - Kyocera](#)
4007 Hiroyuki Sato - Canon
4008 Bob Setterbo - Adobe
4009 Gail Songer, EFI
4010 Mike Timperman - Lexmark
4011 Randy Turner - Sharp
4012 William Wagner - Digital Products
4013 Jim Walker - Dazel
4014 Chris Wellens - Interworking Labs
4015 Rob Whittle - Novell
4016 Don Wright - Lexmark
4017 Lloyd Young - Lexmark
4018 Atsushi Yuki - Kyocera
4019 Peter Zehler, Xerox, Corp.

4020 **9. INDEX**

4021 This index includes the textual conventions, the objects, and the attributes. Textual
 4022 conventions all start with the prefix: "JM" and end with the suffix: "TC". Objects all
 4023 starts with the prefix: "jm" followed by the group name. Attributes are identified with
 4024 enums, and so start with any lower case letter and have no special prefix.

4025	—C—	4059	jmGeneralNumberOfActiveJobs.....	78
		4060	jmGeneralOldestActiveJobIndex.....	78
4026	colorantConsumed.....	4061	jmJobIDJobIndex.....	82
4027	colorantRequested.....	4062	jmJobIDJobSetIndex.....	81
		4063	jmJobImpressionsCompleted.....	86
4028	—D—	4064	jmJobImpressionsPerCopyRequested.....	85
		4065	jmJobIndex.....	83
4029	deviceNameRequested.....	4066	jmJobKOctetsPerCopyRequested.....	84
4030	documentCopiesCompleted.....	4067	jmJobKOctetsProcessed.....	85
4031	documentCopiesRequested.....	4068	jmJobOwner.....	86
4032	documentFormat.....	4069	JmJobServiceTypesTC.....	68
4033	documentFormatIndex.....	4070	JmJobSourcePlatformTypeTC.....	40
4034	documentName.....	4071	jmJobState.....	83
		4072	jmJobStateReasons1.....	84
4035	—F—	4073	JmJobStateReasons1TC.....	69
		4074	JmJobStateReasons2TC.....	73
4036	fileName.....	4075	JmJobStateReasons3TC.....	76
4037	finishing.....	4076	JmJobStateReasons4TC.....	76
4038	fullColorImpressionsCompleted.....	4077	JmJobStateTC.....	49
		4078	JmJobStringTC.....	39
4039	—H—	4079	jmJobSubmissionID.....	81
		4080	JmJobSubmissionIDTypeTC.....	46
4040	highlightColorImpressionsCompleted.....	4081	JmMediumTypeTC.....	44
		4082	JmNaturalLanguageTagTC.....	40
4041	—I—	4083	jmNumberOfInterveningJobs.....	84
		4084	JmPrinterResolutionTC.....	43
4042	impressionsCompletedCurrentCopy.....	4085	JmPrintQualityTC.....	42
4043	impressionsInterpreted.....	4086	JmTimeStampTC.....	40
4044	impressionsSentToDevice.....	4087	JmTonerEconomyTC.....	43
4045	impressionsSpooled.....	4088	JmUTF8StringTC.....	39
		4089	jobAccountName.....	54
4046	—J—	4090	jobCollationType.....	61
		4091	jobCodedCharSet.....	53
4047	jmAttributeInstanceIndex.....	4092	jobComment.....	57
4048	jmAttributeTypeIndex.....	4093	jobCompletionTime.....	66
4049	JmAttributeTypeTC.....	4094	jobCopiesCompleted.....	60
4050	jmAttributeValueAsInteger.....	4095	jobCopiesRequested.....	60
4051	jmAttributeValueAsOctets.....	4096	jobHold.....	58
4052	JmBooleanTC.....	4097	jobHoldUntil.....	58
4053	JmFinishingTC.....	4098	jobKOctetsTransferred.....	60
4054	jmGeneralAttributePersistence.....	4099	jobName.....	54
4055	jmGeneralJobPersistence.....	4100	jobNaturalLanguageTag.....	53
4056	jmGeneralJobSetIndex.....	4101	jobOriginatingHost.....	56
4057	jmGeneralJobSetName.....	4102	jobPriority.....	58
4058	jmGeneralNewestActiveJobIndex.....	4103	jobProcessAfterDateAndTime.....	58
		4104	jobProcessingCPUTime.....	66
		4105	jobServiceTypes.....	55

4106	jobSourceChannelIndex.....	55	4129	printerResolutionRequested.....	59
4107	jobSourcePlatformType.....	55	4130	printerResolutionUsed.....	59
4108	jobStartedBeingHeldTime.....	65	4131	printQualityRequested.....	59
4109	jobStartedProcessingTime.....	66	4132	printQualityUsed.....	59
4110	jobStateReasons2.....	52	4133	processingMessage.....	52
4111	jobStateReasons3.....	52	4134	processingMessageNaturalLanguageTag.....	53
4112	jobStateReasons4.....	52			
4113	jobSubmissionTime.....	65	4135		—Q—
4114	jobSubmissionToServerTime.....	65			
4115	jobURI.....	54	4136	queueNameRequested.....	56
4116		—M—	4137		—S—
4117	mediumConsumed.....	64	4138	serverAssignedJobName.....	54
4118	mediumRequested.....	64	4139	sheetCompletedCopyNumber.....	61
			4140	sheetCompletedDocumentNumber.....	61
4119		—N—	4141	sheetsCompleted.....	63
			4142	sheetsCompletedCurrentCopy.....	63
4120	numberOfDocuments.....	56	4143	sheetsRequested.....	63
			4144	sides.....	58
4121		—O—	4145	submittingApplicationName.....	56
			4146	submittingServerName.....	55
4122	other.....	52			
4123	outputBin.....	58	4147		—T—
4124		—P—	4148	tonerDensityRequested.....	59
			4149	tonerDensityUsed.....	59
4125	pagesCompleted.....	63	4150	tonerEcomonyRequested.....	59
4126	pagesCompletedCurrentCopy.....	63	4151	tonerEcomonyUsed.....	59
4127	pagesRequested.....	63			
4128	physicalDevice.....	56			
4152					