Subj: Issues for Job Monitoring MIB, version 0.85, dated 8/9/97

From: Tom Hastings
Date: 8/8/97
File: issues.doc

This file will be our standing list of open and closed issues for the Job Monitoring MIB. The open issues are in the first section, closed issues not yet incorporated into the MIB specification are in the second section, and the closed issues that are in the current specification are in the third section. When an issue is closed, I indicate the resolution and why and move the issue to the appropriate closed section, depending on whether I've also done the editing for it or not.

Revision marks show changes agreed to at the Redmond, 8/6/97 meeting. When I move an issue from Open to Closed, I only show the revision marks of what changed, not the movement itself.

This version of the issues list includes issues that are in the Internet Draft 04 (same as revision 0.84) that need to be resolved.

I've updated the issues with the agreements reached at the JMP meeting, 6/26/97 and the reasons behind the resolutions.

If you object to any of the proposed resolutions to the issues, please send e-mail.

1. Open Issues

The following issues are open:

none.

2. NoneClosed Issues - not yet reflected in the current draft

The following issues have been closed but have not been incorporated in a draft:

none.

3. Closed Issues - reflected in the current draft (0.85)

The following issues have been closed and have been incorporated into the Internet Draft 05 and version 0.85 or earlier:

Issue 1 - Should we add a standard SNMP **RowStatus** object to the **jmJobTable** and **jmAttributeTable**?

Closed: No. We decided to go with simplicity. Also customers have had experience with printers that stop because some software isn't installed or running properly and so prevents the printer from working. Finally, see issue 9 resolution in which no objects are read-write, but implementers can augment this MIB with objects that allow any object in this MIB to be changed by duly authorized users. We can produce another MIB that augments this MIB in the future, if there is interest and a solution to standard means to write objects (and delete rows).

Issue 2 - If we add **RowStatus** to the **jmJobTable**, should we add a **jmGeneralTableOverflowPolicy** object to the **jmGeneralGroup**?

Closed: No, because we did not agree to add RowStatus and we are also concerned about the printer stopping for some configuration reason. Customers that care about accounting should make sure that the accounting applications are running properly, perhaps even with a daemon program that monitors the accounting programs.

Issue 3 - If we add **jmGeneralTableOverflowPolicy** object should it be read-write?

Closed: No, see issues 1, 2, and 9.

Issue 4 - Need to re-draw the job state transition diagram to add the needsAttention state

Tom H will make a table of all possible state transitions, so that it will mean the IETF requirements for plain text in RFCs.

Closed: I added a job state transition table to replace the job state transition diagram. See the Internet Draft 00 and version 0.71. Called the job state: 'processing-stopped'.

Issue 5 - Restore NMS having to access both the server and printer agents (Configuration 2b)?

Yes, with the following understandings: Configuration 2b will show a monitoring application, a server, and a printer. The MIB will be only in the printer, but the monitoring application is also monitoring the server by some other means than the Job Monitoring MIB. The Job Monitoring MIB in the Printer shall have enough information in it for the monitoring application to find the job in the Printer's Job Monitoring MIB that it found in the server (by other means). In such cases, the server usually deletes its copy of the job, but need not. This configuration covers the configuration supported by the HP 5si Mopier private job monitoring MIB when driven from a Novell server.

ACTION ITEM (Tom Hastings, Bob Pentecost): Tom draw up a new configuration 2b and show to Bob before distributing it to the group.

Closed: I added the agreed configuration, but called it configuration 3, since it has similarities to both configuration 1 and 2. See the Internet Draft 00 and version 0.71.

Issue 6 - If Configuration 2b is added to the spec, how does the monitor relate a job in the server and the copy that is in the printer?

Closed: The new Configuration 2b does NOT have a Job Monitoring MIB in both the server and printer; only in the printer.

Issue 7 - If Configuration 2b is added to the spec, add a Boolean General object that says whether this Job Set requires the NMS to contact the printer's agent too?

Closed: No need, since the configuration won't have a Job Monitoring MIB in the server.

Issue 8 - Should we make a new **jmServerGroup** for objects needed by the **serverOnly**(4) and **bothPrinterAndServer**(5) configurations?

Closed: No, we haven't identified any that are server only that can't be put into the jmAttributeTable, so that non-server's need not implement.

Issue 9 - What objects should be read-write, so that the system administrator can set policy?

Closed: No, for simplicity. Also implementers can augment the Job Monitoring MIB with means to write any object. Add a paragraph that indicates that implementers could allow monitoring applications to modify objects, by adding a private table that contains an encrypted password, with date and time mixed in and set in the clear. In addition, the OID of the object to be written and the new value is contained in the table.

Issue 10 - Should we add an object to specify the policy for SNMP Gets for other user's jobs?

Closed: No, authorization and authentication are beyond SNMP and this Job Monitoring MIB.

Issue 11 - Should the policy object for SNMP Gets for other user's jobs be writeable?

Closed: No, we didn't agree to add such an object in Issue 10 and if we had, it wouldn't have been writeable (see answer to issue 9).

Issue 12 - What is the SNMPv1 and SNMPv2 error that an agent shall return if there is no instrumentation for an object?

Closed: There is no such SNMP error. ALL uninstrumented objects in mandatory groups of any MIB should always correctly return 'read-only' static values specified in 'DEFVAL' clauses. 'DEFVAL' is a perfectly good SMIv2 feature intended to cover this situation. Returning ANY SNMP error for ANY object in a mandatory group with a legal instance qualifier (i.e., set of indices) is NOT legal in a literal reading of the SNMPv2 Protocol spec (RFC 1905, page 10, in 'Get-Request PDU' handling). That's what 'shall implement ALL the objects in this group' means! So add DEFVAL clauses to all objects.

Issue 13 - Why didn't the Printer MIB use this SNMP error instead of returning unknown(2) enums?

Closed: The Printer MIB was correct to use these unknown(2) enums, instead of an SNMP error.

Issue 14 - How do we add traps without adding too much network traffic?

Closed: For simplicity and to avoid the design problem of registering and unregistering for traps, we decided not to add traps. The HP 5si Mopier private job monitoring MIB has only one trap: when a job is added to the table. However, no application is using the trap. Polling seems sufficient and not a problem.

Issue 15 - Should **jmGeneralQueuingAlgorithm** be writeable, so that the system administrator using an NMS can change the scheduling algorithm?

Closed: No, we agreed to delete this object, since no implementations of job monitoring with any protocol have such an object. Even Printxchange did not implement this attribute, even though ISO DPA has this Printer attribute.

Issue 16 - Add **passThrough**(6) to **jmGeneralQueuingAlgorithm** for servers that just pass jobs through without queuing?

Closed: No, none of the implementations even had the jmGeneralQueuingAlgorithm object, so we decided to delete the entire object. So we don't need to even decide whether to add a new enum value to it.

Issue 20 - OK to have added **fileName**(3) to **JmAttributeTypeTC**?

Closed: Yes, some implementations have both file names and document names, so we need both.

Issue 21 - Change **physicalDevice**(11) to a text string, so it can be used with servers that don't have the Printer MIB?

Closed: Have both **physicalDeviceName** and **deviceIndex** as resources in the **jmAttributeTable**, so that neither, one, or the other or both can be implemented.

Closed: Issue 22 - Why not require the agent to always return FAX numbers in ASCII, since it is easy to convert from Unicode to ASCII?

Closed: We decided to remove the fax number resource entirely, since it doesn't relate to printing. When a FAX job monitoring MIB is developed to augment this Job Monitoring MIB, it will need other objects, besides the FAX phone numbers. The FAX phone

numbers enum can be registered at that time as a type 2 enum for use with the jmAttributeType object. (Had we kept this enum, and when it is registered, the data type will be ASCII, rather than the coded character set of the implementation), in order to fit into 63 octets. Two-octet Unicode would exceed the space since numbers with password, and extensions, etc., could exceed 31 digits.

Issue 23 - Add resource item to indicate the output-bins that the job requests/uses?

Closed: Yes, it was in several implementations. So add outputBin as a resource enum which can have multiple entries, since some jobs may actually use more than one output bin.

We also agreed to add **colorantConsumed** as a resource enum and **mediumConsumed** where the **jmResourceName** object would be the name of the actual colorant or medium consumed, with one row per different colorant and medium.

Issue 24 - Move any resource items to the **jmJobGroup**, because monitoring applications needs to access the resource frequently without having to read the entire jmAttributeTable?

Closed: No, don't move any. In fact see issue 30, where we agreed to add a fourth index to the jmAttributeTable, which makes all resources directly addressable by jmAttributeTypeIndex as the third index.

Issue 26 - Which indexes shall be persistent across power off and which need not be?

Closed: The persistent indexes shall be: **jmJobIndex** and **jmGeneralJobSetIndex**

Issue 27 - Should **jmGeneralJobCompletedPolicy** be writeable, so that the system administrator using an NMS can change the length of time that completed jobs are kept?

Closed: No, see answer to issue 9.

Issue 28 - Should **jmGeneralQueuingAlgorithm** be writeable, so that the system administrator using an NMS can change the scheduling algorithm?

Closed: No, none of the implementations even had the **jmGeneralQueuingAlgorithm** object, so we decided to delete the entire object. So we don’t need to even decide whether to make the object writeable.

Issue 29 - OK to require that **jmCompletedIndex** be monatonically increasing?

Closed: Yes. Also **jmQueueIndex**.

Issue 30 - Should we move any **jmJobGroup** objects to the **jmResourceGroup**?

Closed: We agreed to add a fourth index to **jmResourceTable**. That index is **jmResourceType** enum, which will actually be the third index. The **jmResourceIndex** will be moved from third to fourth position. See move8obj.doc in the contributions sub-direction.

This makes each resource in the **jmResourceTable** directly addressable by **jmResourceType**. making it as efficient to find an resource in the **jmResourceTable** as to get an object in the **jmJobTable**.

Consequently, we agreed to move the following 9 objects to the **jmResourcesTable**:

1. **jmDeviceIndex**
2. **jmJobSourceChannel - see Issue 37**
3. **jmJobSubmissionTime**
4. **jmJobComment**
5. **jmJobTotalKOctets**
6. **jmJobKOctetsCompleted**
7. **jmJobStartedProcessingTime**
8. **jmJobCompletionTime**
9. **jmJobAccountName**

Issue 31 - Should we re-introduce **jmJobDeviceId** to handle configuration 2b?

Closed: We agreed to bring back configuration 2b in which the NMS has to query the server by means outside the Job Monitoring MIB and the printer using the Job Monitoring MIB. However, in this scenario, there is no Job Monitoring MIB in the server, so there is no need to add back jmJobDeviceId for the job id assigned by the Printer for use in a server's Job Monitoring MIB.

Issue 32 - Shouldn't we require any numeric portion of the client-side identifiers to always be in the jmJobIdNumber object?

The new Job Id table that Harry proposed replaces the jmJobIdNumber and jmJobIdName objects.

Issue 33 - Why have two client-side identifier objects?

Closed: Some job submission protocols, such as BSD LPR/LPD require two.

Issue 34 - What is the SNMPv1 and SNMPv2 error that an agent shall return if there is no instrumentation for an object?

Closed: There is no such SNMP error. See the answer to Issue 12.

Issue 37 - Change the **jmJobSourceChannel** from an index in the Printer MIB to the enum, since the server need not implement the Printer MIB?

Closed: No, spoolers typically know the job source channel. So keep as an index into the Printer MIB for use in Printers. Since we are moving this to the Resource Table, make the enum name reflect the index: jobSourceChannelIndex.

Issue 38 - Do we need to add the **jmJobChannelInformation** object to the new jmServerGroup for servers that don't have a corresponding Printer MIB?

Closed: No, since we agreed to keep the Job Source Channel as an index into the Printer MIB, the monitoring application can access the **jmJobChannelInformation** object in the Printer MIB. see issue 37.

Issue 39 - ISSUE - Why not return the SNMP error ???, instead of -2, if the total K octets is unknown?

Closed: There is no such error. Return unknown(-2) if not known.

Issue 41 - Is it worth rounding down **jmJobKOctetsCompleted** until the job completes and then round up?

Closed: No, lets round up to the next higher K as with **jmJobKTotalOctets**. The only time rounding down could make a difference is for a 1K job and that short a job will happen so quickly that the difference between rounding up versus rounding down can not be seen by people.

Issue 42 - Are interpreters(10), sheetsCompleted(14), processingTime(20) the right resource items to require agents to implement?

Closed: We agreed only to make **sheetsCompleted**(14) mandatory, since the Printer MIB requires it. All other resource items are conditionally mandatory.

Issue 43 - How can **jmResourceName** be a union of OCTET STRING, Integer32, and Counter32?

Closed: We agreed to replace **jmResourceName** by two objects: **jmResourceNameAsText** and **jmResourceNameAsType**. See res-type.doc in contributions sub-directory. We also agreed that each resource fills in either **jmResourceNameAsText** or **jmResourceNameAsType**, but not both. Also, except for **mediaConsumed**, no resource item that fills in a **jmResourceAmount** with a count, also fills in **jmResourceNameAsText** or **jmResourceNameAsType**.

Subsequently we agreed to combine the three objects: **jmResourceNameAsText**, **jmResourceNameAsType** and **jmResourceAmount** into just two objects: **jmResourceValueAsText** and **jmResourceValueAsInteger**.

Issue 44 - There was agreement that the Job Monitoring MIB needs a primary index that can separate jobs into disjoint sets for purposes of scheduling. This primary index serves the same purpose for the Job Monitoring MIB as the hrDeviceIndex does for the Printer MIB, except that the Job Monitoring MIB did not want to require the Host Resources MIB. What is the definition of Job Set? How do job sets relate to queues? Can a Printer have more than one job set without having queuing? For a printer that is fed from multiple external queues, are all the jobs from all those queues in a single job set in the Printer?

Closed: See the Internet Draft 00 (and version 0.71). I added clarifications that an agent in a server (configuration 2) or printer (configuration 1 or 3) will represent each queue in that server or printer as a distinct job set, since the agent is representing the queue that is in the server or printer that the agent is instrumenting. The agent cannot be expected to represent a queue that is elsewhere (upstream or downstream). If the printer is fed from multiple queues from the same or different servers, but the printer has only one queue, then the agent in that printer shall represent that queue in the printer as a single job set.

Though a Job Set is most often representing a job queue, we're not calling the job set and the job set index a queue and a queue index, since a server or printer need not implement a queue and need not have any queuing. See if I succeeded in clarifying the definition of job set in the draft. I added cross references back to the terminology section as well, so that people who skip over the terminology or forget after reading the terminology will be reminded to go back for further explanation.

Issue 45 - After fully agreeing on what a Job Set is, should the name remain Job Set, or be changed to Queue, or Job Pool or Job Group to improve understandability? The new **jmGeneralJobSetName** would also be changed to jmGene**r**alQueueName, or **jmGeneralJobPoolName**, or **jmGeneralJobGroupName**. See the Internet Draft 00 and version 0.71.

Closed: We agreed to leave the name as Job Set.

Issue 46 - Do we need to add a **jmGeneralJobSetName** object so that an operator can determine which job set he/she is looking at. The **jmGeneralJobSetName** is administratively assigned and could be (1) the queue name, (2) the server name if the server has only one job set or (3) the printer name if the printer has only one job set.

Closed: I added the **jmGeneralJobSetName** object to the general table for the same reason that we added **prtGeneralPrinterName** to the Printer MIB. I added the explanation that the Job Set Name can be (1) the queue name, (2) the server name if the

server has only one job set or (3) the printer name if the printer has only one job set in the Terminology section and under each group where the **jmJobSetIndex** is used.

Issue 47 - Should we change the name of the **jmResourceTable** to **jmAttributeTable**, since some of the enums are not resources, such as **fileName** and **documentName**, **jobSubmissionTime**, **jobCompletionTime**, etc.

Closed: I renamed the Resource Group and Table to **jmAttributeGroup** and **jmAttributeTable**. See version 0.71 and the Internet Draft 00.

Issue 48 - Since **jmJobIndex** cannot be 0 according to SNMP rules for indexes, what shall an agent do that is instrumenting a printer or server that uses 0 as a valid job-identifier? Use largest positive integer for job 0? Or the agent map the 0 value to the value that is one higher than the maximum that the server or printer uses?

Closed: With the Job ID table, what the values of the jmJobIndex are relative to job identifiers that the server or printer generate has become less important. However, agents can preserve the same values assigned by the server or printer in the jmJobIndex, by mapping a zero job-identifier value to one higher than the server or printer assigns.

Issue 49 - Should we change the definition of the **jmGeneralMaxNumberOfJobs** to **jmGeneralMaxJobIndex** meaning the maximum value that the **jmJobIndex** object can have and the roll over to 1 happens for the next job received? Or add **jmGeneralMaxJobIndex** as another object in the General table? Then the monitoring application would know what the roll over limit would be. For agents that instrument servers or printers that use a job identifier of 0, the actual maximum number would be one more than the actual job identifier that the server or printer generates. So for LPD, the value of **jmGeneralMaxJobIndex** would by 1000, not 999.

The current definition of **jmGeneralMaxNumberOfJobs** is:

```
jmGeneralMaxNumberOfJobs OBJECT-TYPE
    SYNTAX      Integer32(0..2147483647)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The maximum number of queued and completed jobs
        that this server or print can support at the same
        time.

        The value (-1) indicating other shall indicate that
        there is no fixed limit."
    ::= { jmGeneralEntry 4 }
```

What is the purpose of **jmGeneralMaxNumberOfJobs** as currently defined?

Closed: No one could make a good case for this object, so we agreed to delete it. Also with the new **jmGeneralOldestActiveJobIndex** and **jmGeneralNewestActiveJobIndex**, an application can discover roll-over when the newest is less than the oldest.

ISSUE 50 - Should we define the **PrtInterpreterLangFamilyTC** used with the **documentFormat(12)** attribute in the Job Monitoring MIB module, since the **PrtInterpreterLangFamilyTC** textual convention is not yet in an RFC? Or should the Job Monitoring MIB IMPORT the **PrtInterpreterLangFamilyTC** from the Printer-MIB to be?

Closed: Since the **documentFormat(12)** is an attribute, not an object, the value is the general **jmAttributeValueAsInteger** and so does not explicitly IMPORT any enum for use as attributes (only as objects). So the Internet Draft 00 (and version 0.71) no longer have **PrtInterpreterLangFamilyTC**, which nicely side-steps the issue that there is no RFC with **PrtInterpreterLangFamilyTC** enum defined.

ISSUE 51 - Should the **jmJobCurrentState** (and **JmJobStateTC**) be changed from a type 2 enum to a type 1 enum, since adding states would have serious impact on released clients? Currently the IPP draft has the **job-state** and **printer-state** attributes defined as type 1 enums (actually they've changed the terminology, but not the concept, to *keyword*).

Closed: No, we will keep as a type 2 enum, in case we need to make an addition, such as follow IPP.

ISSUE 52- Should **JmJobStateReasonsTC** and **JmJobServiceTypesTC** be defined using the RFC 1902 BITS built-in? **JmJobStateReasonsTC** is 540 bits, while **JmJobServiceTypesTC** is only 31 bits.

Closed. Instead of using **BITS** which requires compilers that conform to the 19xx RFC series, define the bit values as part of the DESCRIPTION in hexadecimal in each 32-bit integer. Create four **jobStateReason*n*** (*n*=1..4) and four **JmJobStateReasons*n*** TCs.

ISSUE 53 - Should there be an object that specifies the current default coded character set of the Job Monitoring MIB, so that the client can figure out how to interpret objects of type OCTET STRING that are coded characters, in case the client might not be configured the same as the server or printer? See Section 6 in the Internet Draft 00 and revision 0.71 for the discussion of coded character sets, including the use of Unicode/ISO 10646.

Closed: No do not add such an object. Unlike the Printer MIB, the text strings in the Job Monitoring MIB originate from clients, not from the server or printer. Therefore, monitoring application programs are apt to be configured to support the same character set and language as the job submision clients. Alternatively, a Job Monitoring agent that

also implements the Printer MIB could use the objects in the Printer MIB to indicate the current localization of the Job Monitoring MIB.

ISSUE 54 - Should we move all of the objects in the **jmJobTable** (9 objects) into the **jmAttributeTable** as enums and then specify some of them to be required for implementation? What about the **jmQueueTable** (6 objects)? What about the **jmCompletedTable** (3 objects)? This would reduce the number of required objects from 21 mandatory objects and 6 conditionally mandatory objects to just 9 mandatory objects and no conditionally mandatory objects.

Closed: We agreed to Harry's proposal to move all of jmJobTable objects to the Attribute Table, to delete the Queue and Completed Table and replace them with the jmJobIDTable and the jmJobStateTable.

ISSUE 55 - Should we remove the **needsAttention** state to align with IPP (and DPA)? The **printer-stopped job-state-reasons** value from IPP has already been added to the **JmJobStateReasonsTC**, so that the user will be able to find out that the job that is **processing** has a printer stopped.

Closed: No, we like the **needsAttention** state. Furthermore, it will be straightforward for an agent instrumenting an IPP server or printer, to map the job and printer states, including job-state-reasons and printer-state-reasons, to the Job Monitoring MIB job states.

ISSUE 56 - Which of the **jmJobTable** entries that were moved to the **jmAttributeTable** should be mandatory enums, if any? They were all mandatory when they were in the **jmJobTable**.

1. **physicalDeviceName (or physicalDeviceIndex)**
2. **jobSourceChannelIndex**
3. **jobSubmissionDateAndTime or jobSubmissionTimeStamp**
4. **jobComment**
5. **jobKOctetsTotal**
6. **jobKOctetsCompleted**
7. **jobStartedProcessingDateAndTime or jobStartedProcessingTimeStamp**
8. **jobCompletedDateAndTime or jobCompletedTimeStamp**
9. **jobAccountName**

Or should we move any mandatory attributes, such as **sheetsCompleted** back to the **jmJobTable**, so that the attributes table contains no mandatory attributes, only conditionally mandatory attributes and the **jmJobTable** is the place that we put the mandatory information?

Closed: We agreed that any of the objects in the jmJobStateTable that are duplicates of the jmAttributeTable, shall be mandatory. This gives the following 8 attributes as mandatory:

jobState
numberOfInterveningJobs
deviceAlertCode
jobKOctetsRequested
jobKOctetsCompleted
impressionsRequested
impressionsCompleted
outputBinName

ISSUE 57 - OK to change **jmAttributeTypeIndex** from not-accessible to read-only, so that it can be mentioned in the conformance clause where we specify that **sheetsCompleted** is the only attribute that is mandatory (so far). Currently the Internet Draft and version 0.71 get a compile error when attempting to mention an enum that is mandatory for an object that is not listed in the conformance clause. Objects that are not-accessible cannot be mentioned in the conformance clauses, but read-only can, since they (**jmAttributeTypeIndex**) would also get added to the list of objects in the group.

Closed: No, leave **jmAttributeTypeIndex** as non-accessible. Provide comments to list the attributes that are mandatory in the conformace section.

Issue 58 - OK that sides, **documentFormat**, and **physicalDeviceIndex**/**Name**, remain as the only attributes that are both requested and used at the same time (with the same instance in the **jmAttributeTable**) as suggested in Ron's EMail of 2/15, or should we make four new attributes that are used/consumed and change the current ones to be requested?

Closed: Leave as they are. The difference between requested and consumed is not very great for these attributes.

Issue 59 - Add the name of the source server or client? As an object or an attribute? See Bob Pentecost EMail of 2/25. Need more specific text for such proposals.

Closed: Agreed in principle, but need a specific specification for each.

Issue 60 - Add the "file name of the job" and a "source port object to tell which client port the job came from"? As objects or attributes? See Bob Pentecost EMail of 2/25. Need more specific text for such proposals.

ACTION ITEM (Tom, Bob): Write up a proposal and send to the DL.

Issue 61 - Need to clarify the semantics of each object and attribute with respect to Configuration 1, 2, and 3. See Bob Pentecost EMail of 3/10 (HP internal review). Most objects refer to the jobs as they exist in the server or printer that the agent embedded in, i.e., is instrumenting. A few objects, represent information that comes from upstream

places in the case of configuration 1 from the client, in the case of configuration 2, the client as well, and in the case of configuration 3, the server and maybe even the client as well.

ACTION ITEM (Tom): Analyze the existing attributes to see if the semantics need clarification depending on which configuration and send to the DL.

Closed: No more ambiguous attributes found.

Issue 62 - Harry Lewis has a proposal for a mapping table that allows a monitoring application that knows a client identifier to directly address the mapping table with a single get in order to find the **jmJobIndex** that the printer is using. See 3/5/97 and 3/28/97 EMail and ftp.pwg.org/pub/pwg/jmp/contributions/pwgjm.pdf. Harry will make a presentation at the JMP meeting.

Closed: The JMP accepted Harry's proposal.

Issue 63 - Should we add attributes for inkjet plotters? See EMail from Patrick Powell, of 3/4/97

Closed: Wait until someone comes forward and wants such a value.

Issue 64 - Need to fill out Appendix A on mapping from the job submission protocols to the Job Monitoring MIB for each of the three configurations.

Closed: Put into a separate document.

ACTION ITEM (all): Write up your job submission protocol mapping to the Job Monitoring MIB.

Issue 65 - What Appendices should remain, which should be separate Internet Drafts and/or informational RFCs and which should disappear?

Closed: No appendices for the Job Monitoring MIB, except for supplemental information about the semantics of job states. A second appendix indicates the job submission protocols that support the **jmJobSubmissionID** concept and what mechanism is used for same. Put any other information into a separate informational RFC, such as mapping to ISO DPA, mapping to IPP, mapping to other job submission protocols, etc.

Issue 66 - What attributes need to have Server vs. Printer values, since both may be needed in a single implementation?

From Harry Lewis mail of 2/19. There was a need for an attribute to specify whether the submission time was when the job was submitted to the server vs. the printer, since with

configuration 2 and 3, they could happen at quite different times. So I would think that we would need another attribute.

Presumably a server would always have access to a date and time clock, so would *not* need the TimeStamp form for the server. So should we replace:

1. jobSubmissionDateAndTime
2. JobSubmissionTimeStamp

with:

1. jobSubmissionToServerDateAndTime
2. jobSubmissionToPrinterDateAndTime
3. JobSubmissionToPrinterTimeStamp

If you are implementing just a printer and don't know when the job was submitted to the server, you would not implement the **jobSubmissionToServerDateAndTime** attribute.

What other attributes do we need separate instances for the server vs.

the printer in order to handle configuration 2 and 3?

Closed: Change the two attributes to the three attributes.

ACTION ITEM (Tom): Send to the DL, if find other attributes that are different between the server and the printer.

ISSUE 67 - Delete the three objects in the Job State table that duplicate attributes? **jmJobStateKOctetsCompleted**, **jmJobStateImpressionsCompleted**, and **jmJobStateAssociatedValue**?

An app can get all of these from the AttributeTable directlly: **jobStateKOctetsCompleted**, **jobStateImpressionsCompleted**, and **jobStateAssociatedValue.**

Closed: Delete the duplicates from the Attributes Table instead.

ISSUE 68 - Delete the **Job State Group/Table** all together, since all objects are also duplicated as attributes?

If ISSUE 67 does delete the 3 objects from the **Job State table**, then only the **jmJobState** object remains. But that is also available in the Attribute Table as the **jobState** attribute.

Closed: No, the Job State Table is useful to scan for jobs using Get Next and select the desired columns.

ISSUE 69- Does order of assignment of **JmAttributeTypeTC** enums make any difference?

Would it help if the mandatory attributes were first, so that Get Next would pick them up first when getting the next conceptual row?

Closed: No, can't use Get Next to step through jobs. The requester can specify which attributes using Get, since the agent is now required to materialize each supported attribute when the job is accepted. So the application can supply a number of Gets in a single PDU without fear of a error, once the application has learned which attributes the agent implements.

ISSUE 70 - Add some simple general device alert TC, instead of using the Printer MIB **Alert** Codes.

The **PrtAlertCodeTC** generic values are *not* much good to an end user without knowing which subunit. For example, **SubUnitEmpty** isn't very informative by itself. If an implementation also has the Printer MIB, then a lot more information is available, so a copy of the Printer Alert isn't very useful. If the implementation doesn't have the Printer MIB, then the Printer Alert codes aren't informative enough.

Even worse, the deviceAlertCode(10) is Mandatory, which can't be implemented, if there isn't a Printer MIB also implemented.

Closed: No, use the alert codes.

ISSUE 71 - Are there any attributes that need to be clarified as to which apply to servers and which apply to devices and which apply to either?

ACTION ITEM (Tom): Send to the DL, if find other attributes that are different between the server and the printer.

Closed: no more found.

ISSUE 72 - What should happen to **jmGeneralNewestActiveJobIndex** when all the active jobs complete?

Shall agent set it to 0 or leave it alone as a pointer to increment when the next job is accepted? If it is reset to 0 (to indicate that there are no more active jobs), what remembers where to put the next received job? What is remembered across power cycles, so that jmJobIndex values are not immediately re-assigned upon power up? If the newest active job is comleted before an older one, shall the agent search back to find the newest still active job and decrement **jmGeneralNewestJobIndex** to point to it? Or should this object really be left alone after the newest job completes and be called jmGeneralNewestJobIndex, since the newest job may no longer be active?

Closed: the agent shall reset it to 0 and keep an internal variable for the next row to assign. That internal variable shall be persistent across power cycles. Also the agent shall find the next newest active job, when the newest is canceled or completes and there are still active jobs in the tables.

Issue 73 - Is there a problem with **outputBinIndex** being made mandatory?

If **outputBinIndex** is made mandatory, but an implementation doesn't have the Printer MIB, the agent has to put 0 as the value. Should we add one more attribute: **outputBinNumber**, which is just a number, not an index into the Printer MIB? If we do, which should be mandatory? Just one more reason to get rid of the jmStateTable, which is forcing us to pick a particular outputBin implementation and make it mandatory. If we got rid of the JobState table, we could forget about making any of the 3 outputBinName, outputBinNumber, or outputBinIndex attribute mandatory.

Closed: Don't add outputBinNumber. Don't make outputBinIndex MANDATORY. Also keep **outputBinIndex** as a MULTI-ROW attribute, so don't need to add multi(-3) enum value.

ISSUE 74: Collapse pairs of attributes that use Integer vs Octets valus?

Analysis of the 78 attributes shows that there are 8 attribute pairs that could be collapsed into one attribute with the implementation using either the Integer or the Octets value (or both) to representthe attribute. The application would have to query both value objects.But if it is using GetNext, it has to get both each time anyway. Only if it is directly accessing an attribute would it have to get both values. On the other hand, it would be fewer Gets than having two attributes as we have now. The 8 pairs are:

physicalDeviceIndex	physicalDeviceName
outputBinIndex	outputBinName
mediumRequestedType	mediumRequestedName
colorantRequestedIndex	colorantRequestedName
colorantConsumedIndex	colorantConsumedName
jobSubmissionToDeviceDateAndTime	jobSubmissionToDeviceTimeStamp
jobStartedProcessingDateAndTime	jobStartedProcessingTimeStamp
jobCompletedDateAndTime	jobCompletedTimeStamp

Should we collapse them into the following:

physicalDevice
outputBin
mediumRequested
colorantRequested
colorantConsumed
jobSubmissiontToDeviceTime
jobStartedProcessingTime
jobCompletedTime

Should we allow an implementation to fill in both with meaningful values in a single row?

Closed: Yes, and allow agents to implement one, the other, or both values. Making it easy for an agent to do both will make such an application that doesn't want to depend on

the Printer MIB being implemented, i.e., the application wants to work with all three configurations.

ISSUE 75 - Should the Attribute enum values be grouped so additions could be added in the appropriate section

When producing the first Internet-Draft, I re-arranged the Attribute enums into logical groups, so that attributes would be easier to find. We now have 78 attributes, so logical grouping is becoming important to make the list more understandable. Several people had proposed adding attributes that were already present in the spec. Also Harry has expressed the concern that any re-assignment of at least OIDs, causes problems with tracking the drafts Finally, when the standard achieves proposed status, there will be additional registrations. It might be helpful if the enums could be assigned to the appropriate group, instead of only at the end.

The current logical grouping are:

Job State attributes	10
Job Identification attributes	19
Job Parameter attributes	7
Image Quality attributes (requested and used)	6
Job Progress attributes (requested and consumed)	7
Impression attributes (requested and consumed)	6
Page attributes (requested and consumed)	3
Sheet attributes (requested and consumed)	3
Resource attributes (requested and consumed)	7
Time attributes (set by server or device)	9

Ok to assign Job State and Job Identification in steps of 30 and the rest in steps of 20?

See also Issue 69. We could put the mandatory attributes first, and then group the rest as above.

Closed: Yes, group the enum assignments.

Issue 76 - So should **jobName**, **jobOwner**, and one of **deviceNameRequested** or **queueNameRequested** be made Mandatory?

When we moved attributes from the job table to the attributes table (Issue 54 and 56), we didn't make any of them mandatory for an agent to implement. Should any of them be made Mandatory?

The old job table had the following (mandatory) objects in it:

jmJobName
jmJobIdName
jmJobIdNumber
jmJobServiceType
jmJobOwner
jmJobDeviceNameOrQueueRequested

jmJobCurrentState
jmJobStateReasons

jmJobIdName and **jmJobIdNumber** have been replaced by **jmJobSubmissionIDIndex** which is Mandatory.

jmJobServiceType need not be Mandatory.

Also **jmJobDeviceNameOrQueueRequested** has been made into two separate attributes: **deviceNameRequested** and **queueNameRequested**, so we'd have to make either one of them mandatory.

jmJobCurrentState is now **jobState** and is Mandatory

jmJobStateReasons became four attributes: **jobStateReasons1, jobStateReasons2, jobStateReasons3,** and **jobStateReasons4.** None of them need to be Mandatory.

So should **jobName**, **jobOwner**, and one of **deviceNameRequested** or **queueNameRequested** be made Mandatory?
Closed: Only jobOwner is made manadatory, but it will remain in the Attribute table, rather than being moved to the Job table.

Issue 77 - Should **jobCompletedDateAndTime/TimeStam**p be canceled time too, or add **jobCanceledDateAndTime/TimeStamp**?

Should we just clarify the **jobCompletedDateAndTime** and **jobCompletedTimeStamp** attributes may be used for either the time that the job completes or the time that the job canceled?

Or is it better to add two new attributes: **jobCanceledDateAndTime** and **jobCanceledTimeStamp**?

Closed: Clarify that completed included canceled and aborted.

Issue 78 - Should the "multiplexor" **jobStateAssociatedValue(4)** attribute be removed from the Job Attribute Table and the equivalent **jmJobStateAssociatedValue** object be removed from the Job State table?

The associated values are also available as attributes in the attribute table. The application has to either (1) request all 7 associated attributes or (2) first request the jobState(3) attribute and the request the 1 pertenent attribute. Since all 7 will easily fit in a PDU (minimum of 500 octets or so on all systems) and each request takes about 20 octets, so you can get about 20 (5*4) attributes into a single PDU.

Closed: Yes. The application can request each object and/or attribute directly and it will fit into a single PDU (20 objects or attributes).

Issue 79 - Should the **'printing'** state be combined into the **'processing'** state?

Many printers don't distinguish between '**processing'** and **'printing'**, especially desktop printers. For those that do, having a state change that really reflects progress, such as the transition from processing to printing, is better handled as a job state reason, not as a fundamental state change. Finally, since this MIB is intended for non-printing services in the future, such as fax out, CD-ROM writing, fax-in, scanning, etc., it would help if one of the states wasn't **'printing'**. Even IPP, only has the state of **'processing'**, with a job-state-reason of '**job-printing**' for those implementations that make the distinction and want to go to the trouble of indicating the difference. IPP even indicates that "most implementations won't bother with this nuance".

Closed: Yes, but the other differences between JMP and IPP need discussion with IPP.

Issue 80 - How handle IPP "**sides**" attribute?

The IPP "**sides**" attribute shows more than just 1 or 2. It has the following values: '**1-sided**', '**2-sided-long-edge**' (i.e., duplex), and '**2-sided-short-edge**' (i.e., tumble). Alternatives:

(a) Change the Job Monitoring MIB "**sides(32)**" attribute to an enum with these three values.

(b) Add a new sides attribute, say, "**sidesType**" with these 3 enum values, and keep the current "**sides(32)"** as an Integer32(1..2) (actually Integer32(-2..2), so can represent other and unknown).

Closed: Don't include the IPP values; the agent can map them to 1 or 2.

Issue 81 - Add IPP "numberUp" attribute?

One of the IPP attributes that might be helpful to an administrator would be to record number-up. An administrator that is bent on saving paper, might give rewards (or lower charges) to users that used number-up.

Closed: No. Can get whether number up is being used by comparing the conditionally mandatory **pagesCompleted** attribute with the **jmJobImpressionsCompleted** object.

ISSUE 82 - Change the OID assignment as Jeff Case and David Perkins suggest:

```
>         jobmonMIB
>         jobmonMIB.1 jobmonMIBObjects
>         jobmonMIB.1.1 jmGeneral
          jobmonMIB.1.1.1 jmGeneralTable
          jobmonMIB.1.1.1.1 jmGeneralEntry
          jobmonMIB.1.1.1.1.1 jmGeneralJobSetIndex
          jobmonMIB.1.1.1.1.2 jmGeneralNumberOfActiveJobs
          jobmonMIB.1.1.1.1.3 jmGeneralOldestActiveJobIndex
          jobmonMIB.1.1.1.1.4 jmGeneralNewestActiveJobIndex
          jobmonMIB.1.1.1.1.5 jmGeneralJobPersistence
```

```
            jobmonMIB.1.1.1.1.6 jmGeneralAttributePersistence
            jobmonMIB.1.1.1.1.7 jmGeneralJobSetName

>           jobmonMIB.1.2 jmJobID
            jobmonMIB.1.1.1 jmJobIDTable
            jobmonMIB.1.1.1.1 jmJobIDEntry
            jobmonMIB.1.1.1.1.1 jmJobSubmissionID
            jobmonMIB.1.1.1.1.2 jmJobIDJobSetIndex
            jobmonMIB.1.1.1.1.3 jmJobIDJobIndex

>           jobmonMIB.1.3 jmJob
            jobmonMIB.1.1.1 jmJobStateTable
            jobmonMIB.1.1.1.1 jmJobStateEntry
            jobmonMIB.1.1.1.1.1 jmJobIndex
            jobmonMIB.1.1.1.1.2 jmJobState
            jobmonMIB.1.1.1.1.3 jmJobStateReason1
            jobmonMIB.1.1.1.1.4 jmNumberOfInterveningJobs
            jobmonMIB.1.1.1.1.5 jmJobKOctetsRequested
            jobmonMIB.1.1.1.1.6 jmJobStateKOctetsProcessed
            jobmonMIB.1.1.1.1.7 jmJobImpressionsRequested
            jobmonMIB.1.1.1.1.8 jmJobStateImpressionsCompleted
            jobmonMIB.1.1.1.1.9 jmJobOwner

>           jobmonMIB.1.4 jmAttribute
            jobmonMIB.1.1.1 jmAttributeTable
            jobmonMIB.1.1.1.1 jmAttributeEntry
            jobmonMIB.1.1.1.1.1 jmAttributeTypeIndex
            jobmonMIB.1.1.1.1.2 jmAttributeInstanceIndex
            jobmonMIB.1.1.1.1.3 jmAttributeValueAsInteger
            jobmonMIB.1.1.1.1.4 jmAttributeValueAsOctets

            jobmonMIB.2     jobmonMIBNotifications  -- reserved

>           jobmonMIB.3    jobmonMIBConformance
            jobmonMIB.3.1  jobmonMIBCompliance
            jobmonMIB.3.2  jmMIBGroups
            jobmonMIB.3.2.1 jmGeneralGroup
            jobmonMIB.3.2.2 jmJobIDGroup
            jobmonMIB.3.2.3 jmJobGroup
            jobmonMIB.3.2.4 jmAttributeGroup
```

Correct?
>yes, if
> 1. jobmon is not somewhere under printmib
> 2. you don't have something like
> jobmonMIB.2 jobmonNotifications
> in which case then
> jobmonMIB.3 jobmonMIBconformance
> would move down one arc

Closed: Yes, including reserving an OID for traps, in case we need them in the future.

Closed: Add **jmGeneralJobSetIndex** to **jmGeneralTable** and **jmJobIndex** to **jmJobTable** to follow SMI rules, even though previous version compiles cleanly. This will avoid what David Perkins refers to as "illegal" indexing. Then have to rename the two columns in the **jmJobIDTable** to not conflict.

ISSUE 83 - Can some attributes be deleted before the **jmGeneralAttributePersistence** expires?

Harry Lewis' 5/2 e-mail suggested that some of the attributes, such as "**numberOfInterveningJobs**(9)" don't even need to persist the shorter time specified by **jmGeneralAttributePersistence**.

Closed: No. All attributes shall be instantiated at the same time and deleted at the same time. Then applications can requrest any number of objects and attributes in a single PDU and not get an error back on one that has been implemented but hasn't been put in the table. The values may change at any time.

Revisited: Subsequently, we relaxed this requirement, so that the agent NEED NOT materialize all supported attributes when the job is received.

ISSUE 84 - Change Associated Value for **'printing'** state to **impressionsCompletedCurrentCopy**(56)?

The MIB attribute jobStateAssociatedValue(4) specifies that the associated value for the 'printing' state is the STATIC attribute: **impressionsRequested**(54). Should we change it to the DYNAMIC value: **impressionsCompletedCurrentCopy**(56)? A monitoring application that wants to create a thermometer can read the STATIC **impressionsRequested**(54) attribute once from the **jmAttributesTable**.

What about the STATIC **impressionsRequested**(54) associated value that is associated with the **'processing'** state? Leave it or change it to something dynamic, like **jobKOctetsCompleted**(50)?

Closed: Since the AssociatedValue object/attribute is being deleted, this issue is moot.

ISSUE 85 - Break the MIB into a monitoring and an accounting MIB?

Need to agree if the accounting MIB augments the monitoring MIB or vice versa?

Then need to agree on which attributes only apply to the augmenting MIB.

Closed: No. There are too many attributes that are used for both monitoring and accounting.

ISSSUE 86 - Clarify jobCopiesRequested(44) vs. documentCopiesRequested(46)

In order that systems that only support one document jobs and systems that support multiple documents per job, use the same attribute when the job has only one document (most usual case) and multiple copies are being made, need to clarify which attribute to use: jobCopiesRequested(44) vs. documentCopiesRequested(46). Also which to use (or both) when muliple copies of a job are requested when the job has multiple documents as well. Need to map clearly to IPP and other job submission protocols.

Closed: Use **jobCopiesRequested** for single document jobs for both systems that support only one documen t per job and ones that support mujltiple documents. Only use

documentCopiesRequested, when a multiple document job actually specifies that individual documents are to be made copies.

ISSUE 87 - When shall the mandatory attributes appear in the **jmJobAttributesTable**?

Shall an agent materalize all mandatory attributes when the job is submitted, so that a requester can access them all with multiple explicit Gets in a single PDU, without fear of a missing object aborting the PDU? If the mandatory attributes are represented as objects in the **jmJobStateTable**, then it is clear from SNMP rules that the agent shall materalize at least an empty value for each mandatory object (attribute).

Closed: The agent can materialize all the attributes when the job is submitted, including with unknown values, or the agent can materialize the attribute subsequently when the attribute values become known. Management applications need to use GetNext to get attributes that might not be present yet or expect SNMP error codes to be returned.

ISSUE 88 - Add **jmGeneralNumberOfJobsProcessed** object since server or printer was booted.?

Most MIBs have some sort of utlization counter. The Job Monitoring MIB should have one also. Add the object to the **jmGeneralTable**. We assume that this object SHALL NOT be persistent across power cycles.

The DESCRIPTION is proposed to be:

```
jmGeneralNumberOfJobsProcessed  OBJECT-TYPE
    SYNTAX      Integer32 (0..2147483647)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The number of jobs that have completed processing
        for this job set since the server or device was
        powered on."
    ::= { jmGeneralEntry 7 }
```

Closed: Do not add **jmGeneralNumberOfJobsProcessed** object. The number of pages printed is more indicative of load, than the number of jobs, since jobs can be short or long.

ISSUE 89 - Add **jmGeneralAttributesImplemented** object with bits for each attribute implemented?

Instead of an application not knowing which attributes an implementation implements and trying to discover by getting errors, or by always using Get Next, instead of Get, how about adding a **jmGeneralAttributesImplemented** object to the **jmGeneralTable** that has a bit for each attribute implemented.

```
jmGeneralAttributesImplemented  OBJECT-TYPE
    SYNTAX      OCTET STRING(SIZE(0..32))
    MAX-ACCESS  read-only
    STATUS      current
```

```
    DESCRIPTION
        "A bit string indicating which JmAttributeTypeTC
        enum values are implemented.  The value is a
        constant independent of which bits are currenly in
        entries in the jmAttributeTable.  The most
        significant bit of the first octet is assigned the
        value 0 to correspond to enum 0 (not used), the next
        most significant bit of the first octet is assigned
        the value 1 to correspond to enum 1 (other), the
        next bit is assigned the value 2 (unknown), 3
        (jobStateReasons2) etc. up to 32*8 - 1 = 255"
    ::= { jmGeneralEntry 8 }
```

Closed: Do not add **jmGeneralAttributesImplemented** object. The representatives of the applications did not think such an object would help. Their applications have to use GetNext anyway, since not all supported attributes are materialized when the job is accepted.

ISSUE 90 - The (MANDATORY) OCTET STRING objects should have a minimum MAX size required

Otherwise, trivial implementations can implement too short sizes and be conforming. The SNMP conformance syntax as the end of the MIB has provision for specifying the minimum maximum that SHALL be implemented. The 63 in OCTET STRING(SIZE(0..63)) is the maximum size and the 0 is the minimum size for an *instance* returned on a Get operation. The (MANDATORY) OCTET STRING objects are with suggested MAX size required:

jmGeneralJobSetName 8 octets required to be implemented

jmJobSubmissionID 32 octets required to be implemented

Closed: Agreed to 8 and 48 as the minimum maximum number of octets that an agent MUST support. We also agreed that jmJobSubmissionID MUST be fixed length, so that an application can omit trailing octets, in order to achieve a "search" capability on just the more significant part of the ID.

ISSUE 91 - The MANDATORY **jobOwner** attribute should have a minimum MAX size required

Otherwise, trivial implementations can implement too short sizes and be conforming.

We suggest that 16 octets of the maximum 63 be required to be implemented. Such a maximum will also help with the next issue.

Closed: We agreed to 16 octets be the minimum maximum that an agent MUST implement.

ISSUE 92 - The MANDATORY **jobOwner** attribute needs to persist as long as there is job data

92a: The MANDATORY **jobOwner** attribute needs to persist as long as there is job data. Otherwise, a client that does not have access to the **jobSubmissionID** or a system that does not have a **jobSubmissionID** cannot identify the jobs in the **jmJobTable**, after the agent removes the job's attributes from the **jmAttributeTable**.

Closed: Agreed.

92b: If it is agreed that the MANDATORY **jobOwner** SHALL persist for the longer period of time, then it should be moved to the **jmJobTable**, where all the other MANDATORY attributes have been made objects.

Closed: Agreed.

92c: Then the **jmAttributeTable** could be made OPTIONAL, so that really low end printers would not need to implement the **jmAttributeTable** at all. Experience at Xerox with low end non-queuing printers suggests that not requiring the **jmAttributeTable** is a win. With a required maximum of only 20 octets (see previous issue), it is reasonable to move the **jobOwner** to the MANDATORY **jmJobTable**.

Closed: No. The **jmAttributeTable** shall be MANDATORY for consistency. There isn't really any difference between an implementation that doesn't implement any attributes and one that doesn't implement the **jmAttributeTable**. So keep the conformance requirements simpler to understand and state.

ISSUE 93 - The **jobName** and **jobSubmission[ToDevice]Time** should be MANDATORY

The windows queue monitoring show three attributes: **jobOwner**, **jobName**, and start time. In IPP, **jobName** is a MANDATORY attribute.

Closed: The **jobName** and **jobSubmissionTime** attributes are NOT MANDATORY. They are like any other attributes that shall be implemented, if the service or device has the functionality and the agent is able to access it.

93a: In order to work for all configurations, the **jobSubmissionToDeviceTime** should be changed to **jobSubmissionTime** and be used for all three configurations: configuration 1: device, configuration 2: server, and configuration 3: device. The **jobSubmissionToServerTime** shall only be used in configuration 3, where **jobSubmissionTime** is also used (for the device). Then **jobSubmissionTime** can be made MANDATORY (since it can be used in all three configurations).

Closed: Agreed.

93b: If the **jobName** attribute is made MANDATORY then it should have a minimum maximum value specified for conformance. We suggest that 12 octets is sufficient.

Closed: Not MANDATORY, so no minimum maximum value specified.

93c: If **jobName** and **jobSubmissionTime** are made MANDATORY, then they should be moved to the **jmJobTable** as well, so that the **jmAttributeTable** can remain OPTIONAL (see previous issue).

Closed: Not MANDATORY, so not moved.

ISSUE 94 - Are the 8 octet fields in the **jobSubmisionID** printable or binary?

The text says "8-decimal-digits". Could it be allowed to be a binary sequence number or random number? Then the chances of collision are even lower.

Closed: printable ASCII.

ISSUE 95 - When reducing the size of the **jobSubmissionID** field from 2 to 1, the other fields weren't increased by 1.

Closed: Fix and increase the jmJobSubmissionID index object from 32 octets to 48 octets, in order to be more unique by reducing the chances of truncation.

ISSUE 96 - Add a **jobSubmissionID** format for **jobOwner**

The first octet would be an ASCII '4'. The next 8 would be a sequential number, and the remaining 23 octets would be the low order 23 octets of the **jobOwner**.

Closed: Agreed.

ISSUE 97 - Add some **jobSubmissionID** formats for numeric identifiers

97a - Add one for POSIX user numbers

Closed: Agreed

97b - Add one for user account numbers

Closed: Agreed

97c - Add one for DTMF incoming FAX routing number

Closed: Agreed

ISSUE 98 - The sequence number and random number in the **jmJobSubmissionID** should be the least significant field, not the most significant field

Then a requester can leave off the sequential number or random number in a GetNext and find all of the jobs from a particular MAC address or client URL (or for a particular **jobOwner**). In order to make this switch, we need to specify that when the MAC address, client URL, (jobOwner, or numberic id) is shorter than 23 octets, that the field is

shortened, rather than being padded out to 23 octets. The least significant field is always 8 octets with leading zeroes, so that we don't need any delimiters between the two fields.

So the spec would become:

```
  Format
  Number   Description
  ------   ------------
  1        octets 2 to n: upto last 23 bytes of the jobName
           attribute; n < 26
           octets n to n+7:  8-decimal-digit random number

  2        octets 2 to n: Client MAC address; n < 26
           octets n to n+7:  8-decimal-digit sequential
           number


  3        octets 2 to n: last 23 bytes of the client URL;
           n < 26
           octets n to n+7:  8-decimal-digit sequential
           number

  ..       to be registered according to procedures of a
           type 2 enum.  See section Error! Reference source
not found. on page Error! Bookmark not defined..
```

Closed: Agreed, but extend the total length from 32 to 48 octets.

ISSUE 99 - The **jobSubmisionID** format '0' makes no sense

99a: There are two interpretations of the text there:

1. The agent only puts a single digit of '0' for the entire jobSubmissionID index - but each subsequent submission would replace the entry.

2. The agent tacks on whatever it wants after the '0' to make a unique jobSubmissionID index for each job.

98b: If interpretation 2 is correct, then could we require the agent to use the **jobOwner** instead, rather than leaving it unspecified how the agent fills in the entry if interpretation is correct?

Closed: The agent SHALL use any of the registered formats when the submitting client does not supply a **jobSubmissionID**.

ISSUE 100 - The **jobSubmissionIDTable** should have **jmJobSet** and **jmJobIndex** as indexes

The current formats will have collisions of **jobSubmissionIDs** occasionally. The statement on lines 1695-1697: "None-the-less, collisions could occur, but without bas consequences, since this MIB is intended to be used only for monitoring jobs, not for controlling and managing them." is incorrect, since future IETF and priviate MIBs are

likely to have 'missles', not 'cameras'. We've had experience with a table like this (**jobClientIdTable**) for a year and a half and we added the **jmJobIndex** equivalent to the row entry that the agent adds to make sure that no entry ever gets overwritten.

So we propose that the **jmJobSet** object and the **jmJobIndex** objects be added as the least significant indexes to the **jmJobSubmissionIDTable**. They are only simple integers and **jmJobSet** is likely to be 1 in most implementations.

Closed: Do not add **jmJobSet** and **jmJobIndex** as indexes to the **jmJobIDTable**. If implementors want to guarrantee uniqueness, they can request to register a new format in which the agent supplies some number of the least significant octets (same as if the submitting client did not supply a **jobSubmissionID**).

ISSUE 101 - add **jobSubmissionID** as an attribute, so can find the ID when scanning a job or attribute table.

An accounting program that wants to find the **jobSubmissionID** would have to scan the entire jmJobSubmissionIDTable

Closed: No, if a management application really needs the **jobSubmissionID** and doesn't know what it is apriori, then that application can scan the **jmJobIDTable** looking for the **jmJobIndex** value that matches.

ISSUE 102 - Make a TC out of the **jobSubmissionID** formats, so can publish new ones more easily?

Closed: Yes.

ISSUE 103 - Specify a minimum required persistence time for **jmGeneralAttributePersistence**

Put the lower bound right in the ASN.1. We suggest 60 seconds as a minimum with a recommendation of 300 (5 minutes). Even add a DEFVAL of 300 as the default. The real low cost device that doesn't want to keep job information around will have a small number of jobs anyway, since how many jobs can just a device process in a minute anyway?

The spec would become:

```
    SYNTAX      Integer32(60..2147483647)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The minimum time in seconds for this instance of
        the Job Set that an entry will remain in the
        jmAttributeTable after processing has completed ,
        i.e., the time in seconds starting when the job
        enters the completed, canceled, or aborted state.
        The value of this object MAY be either (1) set by
```

```
        the system administrator by means outside this
        specification or MAY be (2) fixed by the
        implementation, depending on implementation.

        This value SHALL be equal to or less than the value
        of jmGeneralJobPersistence.  This value SHOULD be at
        least 300 which gives a monitoring application five
        minutes in which to poll for job data."
    DEFVAL      { 300 }          -- 5 minutes
    ::= { jmGeneralEntry 5 }
```

Closed: but the agreed values are 60 (one minute) as a recommendation and 15 (15 seconds) as the minimum, instead of 300 and 60, respectively.

ISSUE 104: Add deviceAlertIndex attribute which is index into alert table?

deviceAlertCode (6) needs pointer to SNMP alert table - See pg. 36. When the device is a printer, the alert code SHALL be the printer alert code. This is the current definition. But, this is not very effective when genericAlertCodes are used. An index into the alert table would provide more information (rather than just JAM, you'd know jam in Input 3, for example). Maybe this is too much info for job monitoring? But it's just as easy for the agent.

Proposed new attribute:

```
deviceAlertIndex(8)     -- Integer32(0..2147483647)
   -- INTEGER:  MULTI-ROW:  The device alert table index for the device that
   -- the job is using.  When the device is a printer, this index SHALL be the
   -- index into the prtAlertTable defined by the Printer MIB[1].  Whether
   -- this attribute is instantiated for this job when another job is
   -- using the device depends on implementation.
```

Closed: No, do not add **deviceAlertIndex**. Also remove **deviceAlert(7)**, since an application can access the Printer MIB if implemented. Also it is problematic for attributes to go away during the life of the job, such as the alert code attribute. Also its not good to duplicate information between two MIBs, since the information can become stale.

ISSUE 105: Ok to clarify that serverAssignedJobName(22) can be all digits?

What happened to **serverAssignedJobNumber** (2x) - See pg. 37

We used to have **serverAssignedJobNumber**, with syntax integer. I think we combined this with **serverAssignedJobName** (22) and dropped it, but in so doing, it is not listed as Octets (only). What about the original concern that (OS/2 and perhaps other) some os's use an integer not a text string. Are we saying the integer must be converted to text?

I think you are referring to the **jmJobIdNumber** attribute that Ron proposed along with jmJobIdName. We deleted both when switching over to your jmJobSubmissionID table

scheme. Ron is in agreement I believe with not needing either jmJobIdName or jmJobIdNumber.

So for servers that assign numbers to jobs before submitting them to devices (configuration 3), rather than names, I would suggest that having the agent converting a number that it received in the job submission protocol to a text string would mean that we could use the same attribute and that an application would not need to deal with two attributes when querying.

However, we should add a sentence clarifying that the text string may be a name or a number.

Closed: Agreed, that **serverAssignedJobName** can be all numbers or even a URL.

ISSUE 106: Should **serverAssignedJobName** persist longer too?

Persistence of serverAssignedJobName - See pgs. 36, 37

Two other attributes (jobOwner and jobName) mandate "long" persistence. If you read the note under serverAssignedJobName, it leads in with the same reasoning, but stops short of requiring "long" persistence. Which persistence value is serverAssignedJobName intended to follow?

Closed: No.

ISSUE 107: Ok to remove the three IPP **timeSinceXxx** attributes?

Eliminate "timeSince" Attributes - See pg. 47 This is too much work for the agent and is contrary to SNMP in that sysUpTime should do the trick. I don't mind using a JmTimeStampTC rather than sysUpTime so much, but the NMS, not the agent, should calculate the times since.

Good issue. The three timeSinceJobWasSubmitted(192), timeSinceStartedProcessing(195), and timeSinceCompleted(197) were added becaue this is the way IPP does these time attributes.

On the other hand, is easier as you point out for the agent to use the JmTimeStampTC jobSubmissionToDeviceTime(191), jobStartedProcessingTime(194), and jobCompletedTime(196) and the NMS can do the work. Also having fewer time attributes to choose from does make the NMS's job easier. I'm in favor of removing the three timeSinceXXX attributes. An agent instrumenting an IPP system will have to do a little more computation.

An agent instrumenting an IPP implemenation will either have access to the time that the job's state change happened and can convert to JmTimeStampTC or only has the time-since-xxx value and will have to substract that from the time of day and substract sysUpTime from the result to return the jmTimeStampTC value.

Closed: Yes, remove the three timeSinceXxxx attributes. They can be computed from other time attributes which are also easier for the agent to deal with since the value does not change with time.

ISSUE 108: Add IMPLIED to **jmJobIDEntry** INDEX statement on page 61?

IMPLIED/IMPLICIT - See pg. 61 The note reads "an IMPLICIT statement is NOT provided in the following INDEX clause, since it was not an SMIv2 feature. Therefore, the extra ASN.1 tag SHALL be included in the varbind in the SNMP request and the response." First, we think the terminology is IMPLIED, not IMPLICIT.

Closed: No. Change jmJobSubmissionID (index) to fixed length, so that a management application can omit trailing octets and get a partial match with SNMPv1 and SNMPv2.

ISSUE 109: Ok for agent to supply defaults for the job attributes depending on implmentation?

"Requested Attribute" defaults For requested attributes like copies, toner, quality etc. what if the requested value is not passed in? Should the agent use the device default?

This is a difference between ISO DPA and IPP. In DPA, the server does populate the job object with the defaults. In IPP, the server doesn't, so that the defaults are only applied if neiter the requester nor the document PDL supplies the attribute.

Closed: Clarify that requested attributes include values defaulted by the server or printer where they have the same semantics as if the requester had supplied them (and so are requested by the system).

ISSUE 110: Break **jmAttributeTable** into two tables: **jmAttributeAsIntegerTable** and **jmAttributeAsOctetsTable** as suggested by David Perkins? Then an application will need about half as many Get Next operations in order to "harvest" all of the attributes, since there won't be any zero-length string values for attributes that don't have strings and -1 or 1 values representing 'other' for attributes that don't have integer values.

Closed: Do not divide the table into two tables. There is some benefit to walking the table in pairs even though most attributes are either integer or octet string, and not both.

ISSUE 111 (restated): How does an application determine the coded character set for the objects and attributes that the agent generates (that cannot come from the job submitting client)? Raised by David Perkins.

The following 3 objects and attributes are in question:

jmGeneralJobSetName object
processingMessage attribute
physicalDevice (name value) attribute

Closed: Add the **JmUTF8StringTC** for these three object/attributes. See separate issue111.doc .pdf.

ISSUE 112 (re-stated): How does a management application determine the coded character set for the per-job objects and attributes that are returned by the agent whether submitted by the job submitting client or defaulted by the agent when the job submitting client does not supply? Raised by David Perkins.

Closed: Add the JmUTF8StringTC for these objects and add a jobCodedCharSet attribute to indicate the set being used.

ISSUE 113: The big concern [from the Area directors] is that from the user's perspective, jobs can be submitted via serial, parallel, or network connections, and the Job MIB is only going to know about the network connections. Raised by Chris Wellens.

Closed: Add additional text to clarify that the device may be directly connected over a serial or parallel port or networked to the client and/or server.

ISSUE 114: If nested jobs are sent to the printer and all have a JobSubmissionID attached, what does the agent do? When a spooler receives a job, it can put a banner page on the job by wrapping it inside of a job. When this occurs, there can be separate JobSubmissionID's for each job. In fact, if the printer doesn't find a JobSubmissionID on the outer job, it will assign one. When the printer gets to the inner job, it will get the true JobSubmissionID that was attached to the client's job. Raised by Bob 'Pentecost on 4/17/97 and re-raised by Harry Lewis on 7/16/97.

Closed: Add a reference to the Appendix B where PJL use of the job submission ID is included and indicate in Appendix B that the later occurrence of a job submission ID is the one that the agent uses. The Appendix B text is:

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

ISSUE 115: If the agent changes state to CANCEL as soon as it becomes aware of the cancel command (to satisfy the end user), there may still be a page or two in the pipeline that the accounting application would miss if it noticed the state change and performed it's data collection.

So, we suggest using jobStateReasons in this case.

CANCEL - processingToStopPoint

which progresses to

CANCEL - jobCanceledByUser

The question is, since these jobStateReasons are not "mandatory", how do we communicate and agree on this recommendation? In other words, how do we achieve interoperability? Raised by Harry Lewis on 7/8/97.

Closed: Move the **processingToStopPoint** reason from the **jobStateReasons2** attribute to the **jmJobStateReasons1** object immediately after abortedBySystem reason and renumber the ones following. Also recommend using processingToStopPoint reason with the aborted and canceled states. The new text is:

processingToStopPoint **0x4000**

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

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

ISSUE 116: Delete the '**other(1)**' **jmJobState** value? I thought we had covered all possible states with the 7 primary plus the **JmJobStateReasons**. Why do we need other? Did we not accomplish what was claimed?Raised by Ron Bergman on 7/28/97.

Closed: delete the 'other(1)' jmJobState value.

ISSUE 117: What is the difference between Toner Economy (**tonerEconomyRequested** and **tonerEconomyUsed**) and Toner Density (**tonerDensityRequested** and **tonerDensityUsed**)? (see page #51) They appear to be identical from the description (or very very close!) Raised by Ron Bergman on 7/28/97.

Closed: Cut and paste error copied the explanation from one to the other. Change the tonerEconomy to say toner economy. Economy is an enum, density is a number from 1 to 100. Several printers have both, so we need both.

ISSUE 118: Alignment of IPP and JMP. A job monitoring MIB agent providing access to an IPP system should be able to access the same data as is created by IPP. Lengths of text characters should be the same, not different (255 vs 63, respectively). Are there any

other differences that would cause problems. See ietf-ipp.doc .pdf in protomap sub-directory. Raised by Paul Moore on 8/7/97.

Closed: Leave the lengths as they are. Most real IPP implementations will not exceed the 63 length in practice.

ISSUE 119: Is the new 32-bit IPP "job-identifier" now the same as the 32-bit jmJobIndex? Is the maximum range 31-bits: 1 to 2**31-1 in both? Raised by Paul Moore on 8/7/97.

Closed: Clarify that the jmJobIndex is the same as the IPP "job-identifier", if IPP keeps the job-identifier. However, push back since the IPP meeting indicates that maybe this issue isn't settled. Also there was discussion that an agent that is providing access to a device that supports multiple job submission protocols, including IPP, may have a problem using the IPP "job-identifiers", unless the device also assigns the identifiers for the other job submission protocols from the same job-identifier number space.

The following text has been incorporated into section 3.6:

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

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

ISSUE 120: The management app should be able to read an object(s) to get the number of consecutive rows in a "packed" table to put into a single Get, so as to reduce network traffic and decrease the time to get the data. What about falling off the end of the job and attribute tables? Raised by Paul Moore on 8/7/97 after having this trouble with the Printer MIB Alert table.

Closed: The attribute table is by definition sparse, since the fourth index is present. For the other tables, since jobs may be canceled or aborted before completing, there can be holes in the job tables, even though we require that the **jmJobIndex** be assigned incrementally on job acceptance. So we can't add any objects that can help in addition to the **jmGeneralOldestActiveJobIndex** and **jmGeneralNewestActiveJobIndex** that we

already have that say where to start and end. SNMP v2 solves the problem of reducing network traffic by using bulk get.

ISSUE 121: **jmJobKOctetesProcessed** can be a multiple of **jmJobKOctetsRequested**, for implementations that make multiple passes over the data. However, the same difference is not specified for **jmJobImpressionsComplete**/**jmJobImpressionsRequested** objects and for **pagesCompleted**/**pagesRequested** attributes.

Closed: Make **jmJobImpressionsComplete**/**jmJobImpressionsRequested** objects and for **pagesCompleted**/**pagesRequested** attributes consistent with **jmJobKOctetesProcessed**/**jmJobKOctetsRequested.**