# Minutes of the JMP Activities of the PWG in San Diego

# May, 1997

## Overview

The meeting, held Friday, May 16, began with Ron Bergman (Chairman) reviewing the fact that the PWG-JMP group is now, officially, an IETF chartered working group. To date, there have been no comments on our IETF Job MIB draft from anyone other than PWG-JMP members.

Several key decisions were made at the May meeting, many details were resolved and a few issues were left opened. The first main decision was to keep the Job State Table, which had been the subject of a lot of debate. Tom Hastings contributed an excellent study of the use of SNMP GET and GET NEXT operations to traverse both the Job State and Job Attributes tables. Tom's analysis showed that the Job State table, which is indexed by jmJobSet and jmJobSubmissionID, provided the most efficient means of traversing a sparse table of Jobs. The Job State Table was renamed to Job Table and the word "State" was removed from the appropriate objects, simplifying the terminology. The second major decision was to eliminate the jmJobStateAssociatedValue object entirely. This object had been proposed as a result of IBM's prototyping and represented a compact form of associating meaningful values with the Job State Table relative to the current jmJobState. While the object did serve the purpose of keeping the Job State Table compact (desired to facilitate greater persistence of entries in this table), it was deemed too foreign by most members due to it's ability to take on different meaning depending on Job State. As jmJobStateAssociatedValue was removed, several mandatory objects were added to the Job State Table.

*Please note. I have attempted to report the activities of the JMP meeting as accurately as I was able to record or recollect them. In the section titled "Job ID", below, I have taken liberty to edit in 2 alternatives and some additional comments which were not actually brought up at the meeting. I am doing this because, in review, I believe this topic was rushed and poorly focused at the meeting and is in need of clarification and broader consensus.*

## Job Table

The debate over whether or not the Job MIB should have a Job State Table, as proposed by IBM following their Job MIB prototyping was settled in favor of keeping the Job State Table (but renaming it to Job Table) thanks to a very thorough study, by Tom Hastings, of the use of SNMP GET and GET NEXT operations on the perspective Job and Attribute tables. Since the right most index of the Job Table is jmJobIndex (which represents the printer assigned job id and primary index to the attribute table), the Job Table represents the most effective way to traverse a sparse table of job entries.

The jmJobStateAssociatedValue object was eliminated, however, from the Job Table and, in it's place, the following mandatory objects were added:

- numberOfInterveningJobs
- jobKOctetsRequested
- impressionsRequested

The following mandatory objects remain in the Job Table

- jmJobKOctetsCompleted
- jmJobImpressionsCompleted

The following object is mandatory, and desired to be in the Job Table, but would drastically increase the size of the Job Table

- jmJobOwner

The following objects, which were originally part of the Job State Table via association with the jmJobStateAssociatedValue object are now "relinquished" to the Attribute Table for the sake of trying to minimize the number of objects added to the Job Table

- deviceAlert
- outputBinIndex

The resulting Job Table looks like

jmJobSetIndex
jmJobIndex
jmJobState
jmJobKOctetsCompleted
jmJobImpressionsCompleted
jmNumberOfInterveningJobs
jmKOctetsRequested

jmImpressionsRequested

There was enough unresolved debate over the use of OctetsRequested and Completed that I feel this issue may need to be revisited. Also, the fact that jmJobOwner did not make it into the Job Table will upset some and may require additional modifications.

## Shopping Carts & Gas Gauges

The objects jobKOctetsRequested and jobKOctetsCompleted were the focus of much debate. The Requested value has two primary uses.

1. A guide in determining the sizes of jobs that are currently pending in the printer or server.
2. A reference point for the top end of a job progress gauge.

Given these uses, we first agreed that jobKOctetsRequested and Completed should be mandatory job attributes. Next, because we want them available in a “get next job” framework, we agreed that they should both be part of the Job Table. We also agreed that KOctetsCompleted are counted as data is passed to the language interpreter. Typical embedded PDL interpreters do not provide “hooks” for octets actually processed.

Item (1) above is referred to as the Shopping Cart or Check out Line scenario. This is because knowing the number of jobs ahead of you and the relative size of each job in octets is analogous to knowing the length of a grocery checkout line and the amount of goods in each cart. In both cases, the given information is sufficient for making an educated guess but will never guarantee some other line won’t move faster. So, the debate regarding jobKOctetsRequested is whether or not the Job MIB agent should multiply this value by the number of copies requested. The alternative is for the monitoring application to get both attributes and adjust the information displayed accordingly. There was not strong consensus on this issue but the discussion ended with the attributed representing their face value definitions.

## Job ID

The Job MIB consists of 4 groups. These are General, ID, State and Attribute (State has just been renamed to Job but this is not as descriptive). The Job ID Table provides a means to find the jmJobIndex (think of this as a printer assigned job id which is used to index the State and Attribute tables) if you understand and know your Job Submission ID. Several Job Submission ID formats have been registered and HP has even created a PJL command to carry this ID during submission. Some printer drivers or print job port monitor applications will generate these new jobSubmissionIDs and use the new PJL command, but some legacy platforms (UNIX, for example) will continue to rely on the job id information found in LPR/LPD, today. Since the jmJobSubmissionID is limited to 32 octets, and we want to keep it small for the sake of Job ID Table persistence, applications dealing mainly with LPR print submission may not use the Job ID table. There are several alternatives to this situation.

1. An LPR/LPD based jmJobSubmissionID format could be registered. This topic was touched on only briefly, by Jay Martin, but should have received more focus. To fit the current jmJobSubmissionID format size and scheme, someone would still need to associate the header bytes which identify the format as "LPR" and the resulting string would be severely constrained, on the order of the example which follows:
    - Job Owner (6 bytes)
    - Host Name (8 bytes)
    - Server Job ID (16 bytes)
2. A second form of Job ID Table could be introduced specifically for the LPR case. This was also *not* discussed at the meeting but I am including it here for sake of context. We could treat LPR as a special case, give it it's own Job ID Table for compatibility with legacy print subsystems and live with the "old" and "new" forms of Job Submission ID. The LPR Job ID Table would be indexed by the 3 objects noted in (1) above to result in finding the jmJobIndex for use in other tables.
3. The alternative that *was* discussed at the meeting was more of a grope (which is why I took liberty to address alternatives 1 and 2, here) to add one LPR attribute, jmJobOwner, to the Job Table. With this alternative, applications working mainly in an LPR submission domain would abandon use of the Job ID Table and attempt to identify jobs by correlating Owner with jmJobIndex which is returned implicitly on each GET NEXT across the Job Table. This approach not only "ruins" the Job Table by doubling it's size with the jmJobOwner (syntax OCTETs) object, but results in poor utility of the Job MIB for these applications. It seems unlikely that Owner is sufficient to properly identify the job for reliable accounting or monitoring. Extrapolating, it stands to reason that eventually the requirement would grow to encompass adding Host Name and Server Job ID to the Job Table as well. Adding several more strings would totally revise the intended persistence characteristics of the Job Table.

As the topic of job identification in the LPR domain is discussed, I think several things must be kept in mind. First, the Job ID Table and Job (State) Table relationship was designed to take advantage of a new form of concise Job Submission ID. Some support for the Job MIB *must* ultimately be provided in the submission process and jmJobSubmissonID is key to effective print job monitoring. On the other hand, for accounting purposes or providing a "queue view" as in the "shopping cart" scenario, jmJobSubmissionID does not have to enter the picture. Jobs may be correlated with owners, for job credit purposes, using static attributes from the Attribute table. The only operation which really requires direct, immediate correlation of Owner with State is that of notification of job progress or completion. The point, here, is that if an application is performing job state notification to a user, it is likely to also be involved in the job submission process and, therefore, should be able to associate a Job Submission ID. If the application is providing a generic queue view or accounting, the jmJobIndex may be determined by traversing the Job Table and, knowing the index of all Pending, Printing or Completed jobs, the application can gather whatever additional information it needs from the Attribute Table.

*If the design point is to facilitate a job state notification broker that lives outside of the submission framework and correlates users (notification recipients) with print jobs strictly based on identification provided via LPR, we need to have this stated.*

**Nested JobID.**

An issue had been raised by HP, pointing out that, in a QueueServer implementation for example, the NIC card would gets job, wrap the job and create a banner page. This could result in nested Job Submission Ids. Bob Pentecost had proposed 3 alternatives

1. Job MIB agent track innermost JobSubmissionID
2. Job MIB agent track all job IDs and point to same job index
3. Job MIB agent track nested jobs as separate jobs.

Number 2 is the preferred method *(Is this mandatory*?)
*What about other nested attributes*?

## JMP / IPP Comparison

Scott Issacson lead a comparison of JMP and IPP job attributes based off a paper written by Tom Hastings (Tom's paper number 2). Tom has the official disposition of each item on his write-up. Here are some of the highlights which I captured:

1. job-name, medium, job-owner etc. are 255 byts in IPP. They are only 63 bytes in job-mib. OK.
2. n-up - no specific attribute required because the ratio of Pages to Sheets provides all the necessary information for accounting and billing.
3. Sides. Don't need to know binding (long edge, short edge).
4. Printer resolution - Yes we need this attribute
5. Print Quality - Yes we need this attribute
6. Job Originating Host - Yes, this attribute should be added to the Job MIB
7. States comparison - Harry to do proposal.
8. Document formats. Printer MIB vs. MIME. Unresolved

## MISC

1. There was a review of the list of Job Attributes and whether or not we think they apply more to job monitoring or accounting. See Tom's revised list which should be available soon. Some attributes are really useful for identification which applies to both monitoring and accounting. It might be helpful if we also make this distinction.
2. HP has officially submitted a PJL command for the jmJobSubmissionID. Bob Pentecost will document the command and Tom will add it to an addendum on the Internet Draft of the Job MIB. I think the command looks like @PJL JOB SUBMISSIONID = "id string", but wait until you see Bob's documentation to make this "official". Thanks, to HP, for helping standardize the submission of the job ID.
3. It was agreed that we should not duplicate attributes among various tables in the Job MIB. For instance, since the Job Table has a mandatory object called jmJobKOctetsRequested, the Job Attribute Table shall not have an attribute called

jobKOctetsRequested. I don't know what the reasoning was behind this decision other than people felt it was "SNMP foreign" to have two OIDs where the same basic value may be determined. If the IBM Job MIB prototype experience were heeded, however, we would realize a distinction between the *dynamic value* (in the Job State Table) and the *static value* (in the Job Attribute Table) for the *attribute*. These prototypers have tried to distinguish, many times, that the Job State Table is best suited for representing the dynamics of the job in process while the Job Attribute Table is most useful for learning either static information or the value of dynamic attributes which have reached their final state. However, this argument has not been accepted.

4. Since outputBinIndex is no longer part of the (defunct) jmJobStateAssociatedValue, and not part of the Job Table, it becomes an attribute in the Job Attribute Table which means it can easily be multi-valued. Therefore, there is no more need for the previous definition of -2 = multi which was a shorthand way of trying to represent multiple output bins using a single integer.
5. deviceAlertCode also moved from the Job Table to the Attribute Table and can now be multi-valued if necessary There is an unresolved question about what to do when the device uses the generic alert codes. Perhaps, device alert code is too specific for the job MIB in the first place?
6. The enumerated list of job attributes keeps growing and changing. There will always a possibility for the list to grow and we can either tack new attributes to the end of the list or modify the list, now, into some meaningful grouping, leaving space for future expansion. Either approach would be sufficient, architecturally. I believe Tom is favoring the grouped approach. The main goal is to structure the list, one way or the other, such that currently enumerated attributes will not change.
7. It is difficult for an embedded controller to distinguish between Printing and Processing states. Even though Printing would be the intuitive choice, here, the JMP agreed to combine the two states and call them Processing. This choice aligns with IPP which, presumably, has the future potential to encompass more than printing devices (hummm.. Internet *Printing* Protocol?).
8. The statement was made that any supported attribute, whether it resides in the Job Table of the Attribute Table, must be "instantiated" for a particular job as soon as that job has a state of PENDING. We need to clarify if this pertains to *all* supported attributes or just mandatory ones. I think mandatory should suffice since these are all a management application can rely on for reliable varbinds, anyway.
9. It was also stated that no object can be deleted prior to it's tables persistence time-out. For example, even though the jmJobNumberOfInterveningJobs value will reach 0 for every job as it moves from PENDING to PRINTING in the Job Table, the object, itself, must remain instantiated with the final value.
10. There is no need to clarify objects or attributes according to Server based vs. Printer based Job MIB implementations. (Issue 71).
11. jmJobNewestActiveIndex should return to zero when there are no active jobs. There was a proposal to change NewestActive to simple Newest in an attempt to keep the agent from having to "back up" if jobs get completed out of order, but this proposal was rejected. (Issue 72).

12. It was reemphasized that jmJobIndex needs to be monotonically increasing. It is up to the agent to guarantee this.
13. In some cases, to accommodate both Printer and Server based implementations of the Job MIB, where printers would be expected to be able to correlate to the Printer MIB but Servers would not, “sister” attributes have been created. An example would be outputBinIndex and outputBinName. We decided, at this meeting, to “collapse” these “sister” attribute pairs into one, acknowledging that the Job Attribute table already provides for ValueAsInteger and ValueAsOctet.
14. Systems that do not understand multiple documents per job should not implement jmJobDocumentCopies (Requested or Completed). These attributes only pertain to multi-document systems.