Redpaper

Dawn May
Angela Newton
Mike Rusell
Dan Tarara
Kevin Vette

Best Practices for Managing IBM i Jobs and Output (and a few other special tips)

Introduction

Jobs and spooled output consume system resources, mainly the CPU used to create and manage them and the storage to contain them. Typically, in most environments, you can manage resources that jobs and spooled files consume successfully. However, on the largest systems, with the largest numbers of jobs and very large numbers of spooled files, you can encounter limits. Too many jobs can fill up the job table, or too many spooled files can consume all the system's storage. Having a large number of jobs and spooled files in the system can contribute to potentially long IPLs for unexpected outages. Having a very large number of spooled files on a single output queue can result in lock contention.

This IBM® Redpaper publication describes best practices and recommendations for managing jobs and spooled output on IBM i. It provides an overview of the various controls within the IBM i operating system that you can configure to adjust the limits for spooled output and jobs. It also provides recommendations for setting these values.

We do not explain in detail each of the configuration controls that we discuss in this paper. Rather, if you need additional information regarding the topics that we discuss, see the IBM i 6.1 Information Center at:

`http://publib.boulder.ibm.com/iseries/`

Basic considerations for the number of jobs in the system

The maximum number of jobs in the system includes jobs in three basic states:

- Jobs on Job Queues, waiting to run
- Active jobs
- Ended jobs with spooled output

Jobs waiting to run and active jobs must be considered as part of the total jobs on the system. However, jobs that have ended with spooled output do not. You can manage spooled output without using a job to hold it. IBM has found that many IBM i customers can run into limits and resource constraints because they manage spooled output by managing jobs.

You must address jobs that have ended with spooled output by operational changes. Although the IBM i operating system includes several enhancements that provide improved methods to manage spooled output, many customers do not use them. Keep in mind that IBM has no plans to increase the maximum number of jobs that are allowed on IBM i. For this reason, the use of the enhancements that are provided in recent operating system releases alleviates the issue of keeping many spooled files, in particular job logs.

Overview of IBM i jobs and job structures

In this section, we discuss different structures that are involved in IBM i jobs.

The job table

The *job table* is the data structure in which the operating system keeps track of every job in the system. Each job in the system—whether it is on a job queue, active, or ended with spooled output—has one entry in the job table. The maximum number of jobs on the system is limited by the total number of entries that are supported in this job table. However, simply growing the size of this table to accommodate more jobs is not a feasible option because of performance considerations.

The job table can have a large number of unused entries, which can occur in a situation where an unusually large number of jobs are created for some reason (perhaps due to a runaway SBMJOB or `Spawn()` issue). If the job table grows significantly larger than the typical number of jobs on the system, you might want to compress the table. See the next section, “Permanent and temporary job structures” for more details about job table considerations for IPL.

You can use the Display Job Tables (DSPJOBTBL) command to see statistics regarding the job table and the number of jobs in the system.

Permanent and temporary job structures

Every job within the IBM i operating system has data structures to store information regarding the job. These data structures are composed of several components. However, for the discussion in this paper, you need to understand two major components of the job data structures:

- Permanent job structures
- Temporary job structures

A *permanent job structure* is assigned to a job when the job enters the system. This job is not available for reuse until the job ends without spooled output or if all of the spooled output is disassociated from the job, such as by printing it, deleting it, or detaching it. All jobs have a permanent job structure, whether on a job queue, active, or ended with output. The permanent job structures are made up of storage that survives an IPL. In addition, the spool control block and the job message queue are part of this permanent job structure. The spool control block keeps track of spooled files for the job, and the job message queue contains the messages for the job log. Thus, the main purpose of permanent job structures is to hold spooled output for jobs that have ended. Permanent job structure creation is influenced by the QTOTJOB and QADLTOTJ system values.

Temporary job structures do not survive an IPL. The temporary job structures consist of many components, including the process control space and the QTEMP library. For our discussion, you do not need to understand the details of temporary job structures. Temporary job structures are used only by active jobs and are managed using the QACTJOB and QADLACTJ system values.

Both permanent and temporary job structures are recycled for reuse when they are no longer needed.

There are several configuration parameters that control the management of the permanent and temporary job structures on the system. In the next section, we address the configuration parameters that can help you to better manage output.

Best practices for managing output

Managing output is essential to controlling the number of jobs on the system. IBM works with customers regarding issues with having a large numbers of jobs. IBM has found that generally most systems have a relatively small number of active jobs but very large numbers of jobs with spooled output. In fact, the number of jobs that have ended with spooled output is often 10 to 100 times the number of active jobs.

IBM has no plans to increase the maximum number of jobs on the system. For environments where the number of jobs is near the maximum that is allowed, it is essential to change how output is managed, which means using the capability to detach spooled files. This function has existed on the IBM i operating system since V5R2. In addition, V5R4 included enhancements to make it easier to find and manage detached spooled files with the WRKSPLF and WRKJOBLOG commands.

The system values that manage spooled output include:

- **QSPLFACN**: Spooled file action

  Using the QSPLFACN system value and job attribute can be beneficial in keeping down the number of job table entries on the system. If this value is set to *DETACH, when the job ends, the job is removed from the system, and the job table entry is freed. If the value is set to *KEEP (the default), all jobs are kept until their spooled files are all deleted. If running out of job table entries is an issue for you, it is recommended that you set the QSPLFACN (or the job attribute SPLFACN) value to *DETACH. For existing spooled files, you can use the Change Job (CHGJOB) command to remove the job without deleting spooled files that you need to keep.

  One of the consequences of using *DETACH is that the job log (and other spooled files that the job creates) cannot be found or accessed using the job commands (WRKJOB, CHGJOB, HLDJOB, and so forth). However, with V5R4, there are two methods to find spooled files using the job name:

  – The WRKJOBLOG command allows users to find both spooled and pending job logs using the job name (including generics). In addition, the WRKJOBLOG allows a date and time range to further subset the list of available job logs. In V6R1, you can use the F13=Delete all key to delete the entire list of job logs that display.

  – The WRKSPLF command with the JOB option allows the user to show just the spooled files for a specific job or a set of jobs with the same or similar names because generics are again supported.

Another possible issue with using *DETACH is that if you use the same job names over and over, if you use enough jobs, and if your spooled files stay around long enough, you can end up with spooled files with the exact same job name (including the job user and job number), spooled file name, and spooled file number. Older applications that find or work with specific spooled files and only use these parameters to locate a spooled file can have problems with these duplicates. Additional fields added to all spool interfaces allow users to compensate for this situation by specifying the create date and time to get the exact spooled file desired. In addition, all interfaces for finding and listing spooled files return these values so that they can be used programmatically. Older applications might need some updates to handle this situation.

- **QMAXSPLF**: Maximum number of spooled files

  The QMAXSPLF system value sets the maximum number of spooled files that can be spooled under a single job. The default value is 9999, but you can increase it up to 999999. The default value can be useful in limiting a runaway job from producing too many spooled files, but at times certain jobs might need the maximum increased. This situation is especially true if many spooled files on the system are created by server jobs or jobs run under the authority of a swapped to user. In these cases, the spooled files that are generated are placed under a job with the job name of QPRTJOB and the user name of the swapped to user. If these jobs generate a lot of spooled files, you must create a new QPRTJOB each time the maximum set by the QMAXSPLF system value is reached. Reaching the maximum set by the QMAXSPL system value can cause additional jobs to be created, which creates a new job causes a creation of an entry in the job table. Increasing the QMAXSPLF value can allow more spooled files for each of these jobs and can keep the system from creating as many QPRTJOBs.

  If you increase the QMAXSPLF beyond the 9999 default limit, any applications that use the spooled file number to find or work with a spooled file must have a large enough (more than four characters or large enough binary) field to hold the spooled file number. Otherwise, they can fail.

- **QRCLSPLSTG**: Reclaim spool storage

  The QRCLSPLSTG system value controls how long unused database members, which store spooled files, are allowed to stay in the system after the spooled file that uses them is deleted. The allowed values are *NONE, *NOMAX, and a number of days form 1 to 366. This value can make a significant difference in the performance when creating large numbers of spooled files on a system. The value also can limit the amount of space that the system uses due to keeping empty database members on the system, which can be used later to store spooled files.

  If this value is set to *NONE, every time a spooled file is deleted, the database member that stores the spooled file is deleted, and when another spooled file is created, a new database member must be created to store it. While this method minimizes the storage that the system uses to store spooled files, creating the database member can cause performance problems due to the extra time that creating the member takes. It also can cause locking problems on the database files in QSPL.

  Do not set the *NONE value unless you are directed to do so by an IBM support person. If you set the value to *NOMAX, spooled database members are never deleted, even when empty. This value assures that most of the time, a database member is available for the system to use whenever a spooled file is created, thus speeding up the time it takes to create the member and avoiding any locking on the database file in which the member exists. After the spooled file is deleted, the data is cleared and drops the size to about 16 KB per database member. However, with large numbers of database members, system storage usage can be a problem still. A value between 1 to 366 allows the system to delete excess members that perhaps are created due to a runaway job while still allowing a member to be available most of the time.

The best value to use for this system value is the number of days between normal peak usage of spooled files plus one. For example, if you run some weekly reports that generate a lot of spooled files, eight days is the correct value for QRCLSPLSTG. This value allows the weekly peak number of database members (7 + 1) to stay on the system long enough to be available for the next weekly run. However, if for some reason even more spooled files are generated occasionally, the database members that are created to hold these spooled files are deleted after eight days, retuning the number of database members to a normal value.

For more information about this system value and the RCLSPLSTG command, consult the IBM i Information Center.

- **QJOBSPLA**: Spooling control block initial size

  The QJOBSPLA system value controls the initial size of the spool control block that is allocated for each job on the system. In most cases, the default value of 3516 bytes is fine unless the majority of the jobs have more than five inline spooled files, which are the files that are created for inline data in submitted (reader) jobs. This value does not affect the use of spooled output files, and you should not increase it based on the spooled output files that are produced.

- **QADLSPLA**: Spooling control block additional storage

  The QUDLSPLA system value is obsolete and is no longer used.

Additional output management tips

In this section, we discuss additional tips for managing output.

Spooled files on IASPs

Spooled files that are created on output queues in libraries on IASPs are detached automatically from the job. Thus, at end of job time, they are not counted when determining whether the job structure can be freed. This design is necessary because the spooled files must be able to exist if the IASP is varied off and switched to a different system (where the job will not exist). Spooled files on IASPs behave similarly to spooled files that are created for jobs where the spooled file action is *DETACH. They can, however, be found and manipulated by using the job commands (WRKJOB, CHGJOB, HLDJOB, and so forth) until the job ends (or the ASP group is no longer set for the job).

You need to consider other benefits to putting spooled files on IASPs. In V5R4, removing as many spooled files as possible from *SYSBAS and onto an IASP can improve the speed considerably for an abnormal IPL, because the spool rebuilds indexes in *SYSBAS during an abnormal IPL. In V6R1, changes in spool internals allow the spool to not rebuild during the abnormal IPL, so this is not as important. The current limit for the number of spooled files in *SYSBAS is 2 610 000. If your system hits this limit, spooling stops until some spooled files are deleted. The spooled file limit on an IASP is quite a bit higher (currently 10 million).

In addition, putting spooled files in IASPs allows some additional security options. You can restrict certain users to an IASP with the *USE authority and, thus keep others who might need to have *SPLCTL special authority to access all spooled files in *SYSBAS from accessing spooled files in the IASPs that are controlled more stringently.

You also need to consider limitations to putting spooled files in IASPs (in addition to the limitation for spooled files that are detached). For example, if you use WRKSPLF basic assistance level or S/36 format for the display format, the spooled files do not show in IASPs, which can be an issue if some users want the basic assistance level to deal with spooled files. Also, the default output queues for printer devices are not placed in IASPs, so starting a writer requires additional work as opposed to using the WRKWTR option 1. The operator must first make sure that the ASP group is set for the job that starts the writer and then must issue the STRPRTWTR command to specify the output queue on an IASP.

Deleting spooled files automatically

You can control the number of job table entries (and spooled files) on a system by using the support that is provided to expire automatically spooled files that you know you do not need for more than a certain period of time. To make use of this support, you must change the printer files that your applications use to specify an expiration date or a number of days until the spooled files that are created using that printer file expire. The spooled files that are created with an expiration date are not deleted automatically, but instead are considered eligible for deletion at one second before midnight on the date that you set. After that time, they are deleted the next time that the DLTEXPSPLF command runs. To remove the spooled files that expire every day, it is best to set up a job schedule entry that runs once a day at a time when the system is not so busy, as shown in Example 1.

Example 1   ADDJOBSCDE command

```
ADDJOBSCDE JOB(DLTEXPSPLF) CMD(DLTEXPSPLF ASPDEV(*ALL)) FRQ(*WEEKLY)
SCDDATE(*NONE) SCDDAY(*ALL) SCDTIME(010000) JOBQ(QSYS/QSYSNOMAX) TEXT('DELETE
EXPIRED SPOOLED FILES SCHEDULE ENTRY')
```

You can also allow the system to help you manage some system created spooled files, such as job logs and dumps. You can use the CHGCLNUP command to change how long these spooled files are retained. Then, you can use the "Job logs and system output" parameter to specify how long these files are kept on the system. (The default is seven days.) When you use automatic cleanup of job logs and system output, the job logs are spooled to QEZJOBLOG, and dumps are spooled to QEZDEBUG. The entries on these two output queues are removed when they are older than the specified number of days.

Note: Note that this cleanup option also removes pending job logs that meet the criteria for the number of days.

Best practices for managing jobs

System values that manage jobs and job structures include:

- **QMAXJOB**: Maximum number of jobs

  This system value has a shipped value of 163520. You want to set this value high enough so that, if you hit the limit, an error is generated that requires intervention. However, you also want to set this value low enough so that intervention occurs as soon as possible. When the number of jobs reaches this maximum, you can no longer submit or start more jobs on the system. Avoid setting this value to 485000. If you are running close to the maximum number of jobs that the system supports, try to stay at least a few thousand jobs below that upper limit to have some ability to recover if needed. If you have a situation where you run close to the upper limit, an error should occur. Spooled file management is key to keep from running too close to the upper limit of the maximum number of jobs. If you need to keep spooled files, it is recommended that you detach the spooled files from the job to reduce the number of jobs that are consumed just to hold spooled files.

  Warning messages are sent to QSYSOPR as the number of jobs reaches the maximum number of jobs on the system (by default, 90% of the maximum). So, a good setting for this system value is the typical peak number of jobs divided by 0.90, which results in the warning message to QSYSOPR when the jobs have consumed 90% of the maximum.

Support is added to customize when these messages are sent with PTF SI29585 (V5R4) and SI30171 (V6R1). For more details about these PTFs, see “Job table warning” on page 9. You want the QMAXJOB system value and the percentage full setting to result in the CPI1468 message occurring at the correct time so that you can react to the situation in a timely manner.

Runaway usage of the SBMJOB command can consume large numbers of job table entries. You can register and run a SBMJOB command exit program to avoid loops that use job table entries. For more information and a sample exit program, see “SMBJOB exit program” on page 18.

Note: The remaining system values that we describe here (QACTJOB, QADLACT, QADLTOTJ, and QTOTJOB) provide controls for when the temporary and permanent job structures are allocated. For these system values, the recommendations here provide general advice. Generally, detailed analysis of how to set these system values is unnecessary.

- **QACTJOB**: Initial number of active jobs

  This system value is the initial number of temporary job structures that are allocated immediately following an IPL. With V6R1, this system value shipped with a default value of 200. The shipped value in prior releases was 20.

  Set this value to handle the maximum number of jobs that you generally have after an IPL. This system value supports values from 1 to 32767, and setting it under the shipped value of 200 is not recommended. Setting it to a large value lowers the performance of the system for a length of time following an IPL.

- **QADLACTJ**: Additional number of active jobs

  This system value is the additional number of temporary job structures that are allocated when more are needed. With V6R1, this system value shipped with a default value of 30. The shipped value in prior releases was 10.

  When additional job structures are needed, this system value determines how many are created. Although this system value supports values from 1 to 32767, it is not recommended to use very low or very high values. Setting the value too low can result in frequent interruptions to allocate new job structures, and setting the value too high can potentially result in long delays when allocating the job structures.

- **QADLTOTJ**: Additional number of total jobs

  This system value defines the additional permanent job structures that are allocated when there no more available. With V6R1, this system value shipped with a default value of 30. The shipped value in prior releases was 10. Although this system value supports values from 1 to 32767, it is not recommended to use very low or very high values. Setting the value too low can result in frequent interruptions to allocate new job structures, and setting the value too high results in the value being ignored because the system never allocates more than 500 job structures at one time.

- **QTOTJOB**: Initial total number of jobs

  This system value defines the initial number of permanent job structures that are created. With V6R1, this system value shipped with a default value of 200. The shipped value in prior releases was 30. This system value applies only when the job table is rebuilt at IPL, as is done when the user instructs the machine to clear output queues, job queues, and incomplete job logs (*CLEAR/*CLEAR/*CLEAR). Avoid using a high value for this system value. A high value results in a longer IPL, because the system never creates more than 16 384 permanent job structures.

Job table considerations for IPL

The size and management of the job table have IPL considerations. An experience report that discusses considerations for IPL time is available in the IBM i Information Center at:

http://publib.boulder.ibm.com/infocenter/systems/scope/i5os/topic/experience/ipla-3-exprabstract.htm

The following points are key when considering the size and management of the job table:

- Do not compress the job tables with every IPL.

  Compressing the job tables during IPL increases the time that is needed to IPL and also forces the system to create job structures as new work enters the system after IPL. Compress the job tables only if you have an abnormally high number of jobs on the system and have deleted the unneeded jobs.

  Having an abnormally high number of jobs can be caused by runaway job submission or failure in the process of cleaning up old jobs and spooled files. You can use the CHGIPLA command to turn off compression:

  ```
  CHGIPLA CPRJOBTBL(*NONE)
  ```

  If you do need to compress, use CPRJOBTBL(*NEXT) so that the system is not set to compress on every IPL.

- Do not check the job tables every IPL.

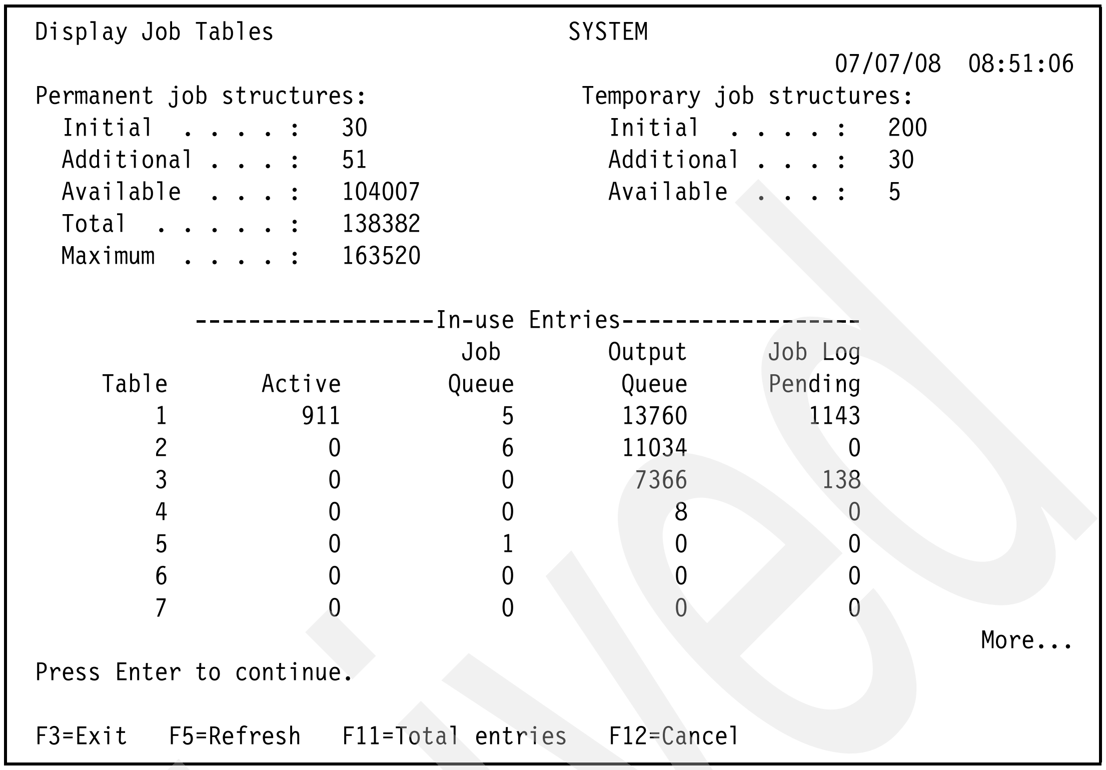
  During an abnormal IPL, all of the jobs in the job tables are deleted and checked for damage. This action can take a significant amount of time during IPL. During a normal IPL, this checking should not be needed. You can turn off job checking during normal IPL with the following command:

  ```
  CHGIPLA CHKJOBTBL(*ABNORMAL)
  ```

Summary

You need to give some thought as to how you set the system values that manage the number of active (temporary) and total (permanent) jobs on the system, remembering that there is a trade-off between storage and performance.

You also need to be aware of the number of jobs that are started during a specific interval of time. When setting these values, the goal is to have enough allocated storage entering the peak usage period so that more job structures are not created when system resources are at their most expensive. Using the DSPJOBTBL command is a good way to monitor this storage to see when the system is close to needing more. The command shows the number of permanent and temporary job structures and whether they are available or in use, as shown in Figure 1.



```
Display Job Tables                      SYSTEM
                                                    07/07/08  08:51:06
Permanent job structures:                Temporary job structures:
  Initial  . . . . :   30                  Initial  . . . . :   200
  Additional . . . :   51                  Additional . . . :   30
  Available  . . . :   104007              Available  . . . :   5
  Total  . . . . . :   138382
  Maximum  . . . . :   163520

            ------------------In-use Entries-----------------
                                      Job         Output      Job Log
     Table        Active            Queue          Queue      Pending
         1           911                5          13760         1143
         2             0                6          11034            0
         3             0                0           7366          138
         4             0                0              8            0
         5             0                1              0            0
         6             0                0              0            0
         7             0                0              0            0
                                                                         More...
Press Enter to continue.

F3=Exit   F5=Refresh   F11=Total entries   F12=Cancel
```

Figure 1 Display Job Tables command

Table 1 summarizes options for reducing the number of jobs in each of the four states.

Table 1 Reducing the number of jobs

Active	**Job Queue**	**Output Queue**	**Job Log Pending**
Reduce the number of idle active jobs. For type: ▶ **Interactive**: See “Managing jobs and job logs with interactive devices” on page 13. ▶ **Writer**: Consider AUTOEND(*YES) on STRPRTWTR. ▶ **Prestart**: Consider tuning prestart job entries to appropriate peak.[a]	If you have a large number of jobs on inactive job queues or held, consider job schedule entries or database files with the //BCHJOB commands.	Delete or detach spooled files. See “Best practices for managing output” on page 3.	Reduce the number of days Job logs and other system output are kept on GO CLEANUP. See “Managing job logs” on page 10.

a. An experience report on tuning prestart job entries is available in the IBM i Information Center:
http://publib.boulder.ibm.com/infocenter/systems/scope/i5os/topic/experience/work2abstract.htm

Job table warning

When the job table is close to full, CPI1468 message is sent to the QSYSOPR message queue. In the past, the message was hardcoded to send this message at 90% full. With V5R4 and V6R1, you can set when this message is sent. In addition to allowing the setting of the threshold for when this message is sent, these PTFs also change how the system decides to send the warning. Prior to the PTFs, the warning was based upon job structure *creation*. With the PTFs, the warning is based upon job structure *use*. Thus, you no longer need to compress job tables (after cleanup) to have the warning message occur when you reach the specified percentage again.

The corresponding PTFs are:

- V5R4 PTF, SI29585: Early Warning for Job Table Full
- V6R1 PTF, SI30171: Early Warning for Job Table Full

Table 2 provides the explanation about how to use them these PTFs.

Table 2 PTF problem summary and problem conclusion

Problem Summary	Message CPI1468 is sent each time the job tables are extended after 90% of the maximum is reached. In some situations, 90% of the maximum is not a sufficient warning.
Problem Conclusion	The PTF allows you to set the percentage to a value lower than 90% to receive a warning earlier. To set the percentage, create a data area named QMAXJOBPCT in the QSYS library with the desired value. For example, to get a warning message at 70%, use: `CRTDTAARA DTAARA(QSYS/QMAXJOBPCT) TYPE(*DEC) LEN(2 0) VALUE(70)` After the data area is created or changed, to force the change to take effect immediately, do one of the following tasks: ► IPL the system ► Change the QMAXJOB system value If the data area does not exist or if the value in the data area is not a 2-digit number in the range of 01 to 89, you continue to receive warning messages at 90%.

Managing job logs

Many environments keep spooled job logs for some period of time for diagnostic purposes. However, keeping job logs, whether as spooled files or in a pending state, consumes a permanent job structure for each job. On some of the largest IBM i installations, these job logs are the major reason for large numbers of jobs on the system. The actual number of active jobs is relatively small compared to the total number of jobs on the system.

IBM does not increase the number of jobs on the system. Detaching spooled files is one method you can use to manage large numbers of spooled files.

For our discussion, we focus on why are job logs kept and whether you can use a more granular control.

First, you can set up a job's logging level to get all messages logged to the job log, but the job log is produced only if the job ends abnormally. If the job ends normally, no job log is generated, using the LOG(4 0 *NOLIST) setting.

When ending jobs, you can indicate that you do not want a job log, as shown in Table 3.

Table 3 Parameter for no job log

Command	Parameter for no job log
End Job (`ENDJOB`)	`LOGLMT(0)`
End Prestart Job (`ENDPJ`)	`LOGLMT(0)`
End Group Job (`ENDGRPJOB`)	`LOG(*NOLIST)`
End Subsystem (`ENDSBS`)	`ENDSBSOPT(*NOJOBLOG)`
End System (`ENDSYS`)	`ENDSBSOPT(*NOJOBLOG)`
Power Down System (`PWRWNSYS`)	`ENDSBSOPT(*NOJOBLOG)`
Clear Job Queue (`CLRJOBQ`)	`LOG(*NONE)`
Sign off (`SIGNOFF`)	`LOG(*NOLIST)`

Consider the following system values with associated job attributes for job log characteristics:

- **QJOBMSGQFL**: Job message queue full action

  This system value determines the behavior when the job message queue is full.

  – *NOWRAP ends the job when the job message queue is full. Use this value to prevent short running jobs from looping endlessly in error situations.

  – *WRAP allows new messages to overlay old messages when the job message queue is full. Use this value for long running jobs.

  – *PRTWRAP creates a spooled file for the job log prior to wrapping to allow new messages to overlay old messages.

- **QJOBMSGQMX**: Maximum size of job message queue

  This system value sets the size of a job's message queue. The shipped value is 16 MB, which for many jobs might be too high. Setting this system value to a high number can over commit a system's storage. For example, if you have 20 000 active jobs and set QJOBMSGQMX to 64, the system can (in the worst case) use 1 280 000 megabytes for job message queues. Even systems that are large enough to run a lot of active jobs probably do not have that much unused DASD. When jobs hit a common error loop, even if it is only 5% of the active jobs, you can start losing storage quickly. Two points:

  – You need to set limits using the system value and the job description values.

  – The system uses storage to run. It uses a lot more storage to handle errors, and you need a fairly wide margin to avoid running out of storage.

  Setting QJOBMSGQMX to 8 and QJOBMSGQFL set to *WRAP is a good, general configuration. If you have a specific need for a larger job log, set these values with care and with knowledge about the jobs that specifically need that high of a job log for special problem cases. Setting the QJOBMSGQMX system value to 8 limits the number of job log messages for a job. It can also reduce the recovery time for IPL after some system failures.

- **QLOGOUTPUT**: Job log output

  This system value allows you to control if, and when, spooled job logs are created.

  – *JOBEND specifies that job logs are spooled when the job ends. Only use *JOBEND if you have a dependency on the job log being available immediately after the job ends. When many jobs end at the same time, using *JOBEND can result in a CPU spike as well as potential contention on the output queue.

  – *JOBLOGSVR specifies that job logs are spooled by the Job Log Servers some time after the job ends. Using *JOBLOGSVR is fairly close to using *JOBEND and tends to operate more smoothly (with less contention and less CPU spike).

    You might want to shut down the job log servers to a restricted state before ending the jobs, depending on how much sequencing you do when ending work. Ending the job log servers prior to going into restricted state prevents the writing of job logs for jobs that are being ended (assuming those jobs use *JOBLOGSVR). IBM generally recommends using *JOBLOGSVR.

  – *PND keeps the job logs in a pending state until they are spooled or deleted from the system. Pending job logs reduce CPU and contention. Pending job logs, however, consume a job structure, even if you are detaching spooled files. The job descriptions for several server jobs are configured to use LOGOUTPUT(*PND), helping to avoid CPU spikes and contention when large number of server jobs end at the same time. The automatic cleanup function (GO CLEANUP) deletes pending job logs in the same manner as spooled job logs; however, if you are not using the system's cleanup feature, then you must do your own cleanup. Cleanup is not optional, because if cleanup is not done, the system can run out of job structures or can fill with spooled files. To clean up pending job logs, use the QWTRMVJL API:

    http://publib.boulder.ibm.com/infocenter/systems/scope/i5os/topic/apis/qwtrmvjl.htm

    Specify a zero value for the "days since job completion" parameter to clean up the job logs immediately. Finally, if you are using the QMHCTLJL API, do not use pending job logs. Using the control job log output API makes sense only if you are creating the job log when the job ends.

  V5R4 introduced the WRKJOBLOG command. You can delete job logs from this interface. In V6R1, you have the option to delete all job logs.

- **QJOBMSGQSZ**: Job message queue initial size

  This system value is obsolete and no longer used.

- **QJOBMSGQTL**: Job message queue maximum initial size.

  This system value is obsolete and no longer used.

Job logs and output queues

If you generate a lot of spooled job logs, particularly at a given point in time, you need to configure the system to avoid contention on the output queue. Because a lock is required to produce a spooled file, you can end up with lock contention when a lot of jobs are generating job logs at the same time. Common scenarios where many job logs can be produced at the same time are network errors that cause server jobs to end or the ending subsystems or the system also creating job logs.

The QEZJOBLOG output log is the main issue for output queue contention because, by default, all job logs go to the one output queue. However, the situation is not limited to QEZJOBLOG if the environment is set up so that the majority of spooled output goes to one or only a few output queues.

For job logs, consider the following approaches:

- As we mentioned previously, avoid creating job logs if possible.
- Use the job log output setting of *JOBLOGSVR to have the job logs produced by the job log servers, which results in less CPU and output queue contention because fewer jobs are creating the job logs at any point in time.
- Configure the system to use multiple output queues for job logs. As shipped, the system has all job logs going to the QEZJOBLOG output queue. You can configure your system to have multiple output queues for job logs. For more information about how to do this configuration, see the article at:

  http://www-1.ibm.com/support/docview.wss?rs=203&q1=qezjoblog&uid=nas1160d62f88ac5547286256a2d0051d855&loc=en_US&cs=utf-8&lang=en

  You need an IBM user ID and password to access this article. You can register for an ID and password at this site.

Managing jobs and job logs with interactive devices

Jobs that are associated with interactive devices (Telnet sessions, display station pass-through sessions, and dumb terminals) have unique considerations.

First, you need to establish how you want device I/O errors handled through the QDEVRCYACN system value or the DEVRCYACN job attribute. Never use *MSG because it allows a different user to connect to the old job. You can use the disconnect options (*DSCMSG or *DSCENDRQS), although some emulators allow different users to connect to old sessions, so caution is advised. The job can be reconnected through the sign-on processing when the same user ID signs on to the same device description. For disconnect processing to work effectively, you need to be using “user-specified named” devices and not the system’s QPADEV* default naming convention.

Overall, the best option is generally *ENDJOBNOLIST, which causes the job to end but does not generate a job log. This option is most beneficial if a network error occurs and a lot of sessions are lost at the same time. In this situation, the creation of many job logs is avoided. If job logs truly are required, the *ENDJOB option is available.

Note: Using the Log Output feature of *JOBLOGSVR or *PND is also valuable for dealing with network errors and losing a lot of sessions at the same time. However, the *ENDJOB and *ENDJOBNOLIST options are the most secure.

If jobs disconnect, the QDSCJOBITV system value, which is the time interval before disconnected jobs end, determines how long these jobs stay in the system in the disconnected state. The shipped value is 240 minutes (4 hours). If users are likely to reconnect to a disconnected job, then keeping the jobs for a reasonable amount of time makes sense.

The QINACTITV (Inactivity Timeout) system value determines how long a display session can remain idle before it times out. When the timeout occurs, the partner system value QINACTMSGQ (Inactive job message queue) determines the action to take. The options are *ENDJOB, *DSCJOB, or the name of a message queue. Using a message queue allows you to implement more granular control of handling the inactivity timeout condition. *DSCJOB is the best way to control the system timing out and to minimize job log creation and management.

You can use the QLMTDEVSSN system value to limit the number of devices to which one user can be signed on at the same time. As of V6R1, this system value allows 1 to 9 devices or an unlimited number. Prior to V6R1, the choices were either 1 or unlimited.

Managing storage

If you have a lot of job logs that are created in a short amount of time, it is possible that the job logs can consume a lot of storage on your system. The following system values can help manage this situation:

- **QSTGLOWACN**: Auxiliary storage lower limit action

  Having some sort of automated reaction to a low storage condition is better than just sending messages. You can use the following command:

  ```
  CHGSYSVAL SYSVAL(QSTGLOWACN) VALUE(*ENDSYS)
  ```

  Note that using the job log servers to generate job logs can also be helpful during critical storage usage scenarios. Using *JOBLOGSVR results in no job logs being generated after the job log servers end. However, if jobs are generated at the end of the job, ending the system ends the jobs, which generates spooled files and which can make the storage situation worse.

- **QSTGLOWLMT**: Auxiliary storage lower limit

  This system value specifies the amount of storage remaining that is available in the system ASP when the storage low action is taken.

  On an ongoing basis, you might want to monitor the storage consumption on your system proactively using one of the following methods:

  – Use the WRKSYSACT command in IBM Performance Tools licensed program product (PT1) prior to V6R1, in the base operating system in V6R1), fourth view (press F11 three times) to show the storage that is allocated and deallocated for the active jobs that display. This information includes both temporary and permanent storage. You can press F19 to refresh this information automatically.

  – Use the WRKSYSSTS command, which has a graphical user interface, to show the auxiliary storage information.

  – Use the WRKDSKSTS command, which has a graphical user interface, to show the percent that each disk unit uses.

  – Use Management Central System Monitors to monitor disk storage. You can monitor the average disk storage and the maximum disk storage. You can find details about Management Central System Monitor metrics at:

    `http://publib.boulder.ibm.com/infocenter/systems/scope/i5os/topic/rzaih/rzaihjobmonmetrics.htm?tocNode=int_222988`

Automatic performance adjustment

The performance adjustment (QPFRADJ) system value controls when the system does automatic performance tuning. The shipped setting of 2 calculates a minimum machine pool size at IPL with no information about the actual workload. As such, it can take several minutes to readjust to the optimal size.

For many systems, it is better to just start with the machine pools size in effect before the system was shut down. You can set up the system to tune performance dynamically but not to perform IPL tuning using the following command:

```
CHGSYSVAL SYSVAL(QPFRADJ) VALUE('3')
```

A common problem for the performance adjuster is reacting quickly enough when there is a dramatic change in workload, such as switching from an overnight batch workload to a daytime interactive workload. The performance adjuster only runs every 1 to 2 minutes, and it can take some time to get a pool to the size that it needs to be.

If you have a significant change in workload characteristics, you can schedule a job to change the shared pool just before the shift change. If the time of the shift change is unpredictable use the CHGSHRPOOL command to set the minimum and maximum size percent (MINPCT and MAXPCT) for the pools.

An experience report on the performance adjuster is available at:

http://publib.boulder.ibm.com/infocenter/systems/scope/i5os/topic/experience/work3abstract.htm

Miscellaneous tips

In this section, we discuss additional tips:

- Before upgrading to a new release, reduce the number of jobs in the system. If the job table has grown significantly larger than the typical number of jobs on the system, you might want to compress the table. A large number of job tables can increase the installation time needed for the upgrade dramatically.
- End jobs, subsystems, and the system using a controlled end with a short delay time (OPTION(*CNTRLD) and DELAY(x)) rather than immediately (OPTION(*IMMED)). When a subsystem is ended in a controlled manner, it stops initiating new work and currently running work gets a chance to complete. When the timeout occurs (or if you do an immediate end), the system ends the running work. It is generally better to not start new work than to start new work that is ended before it completes.
- With V6R1, the PWDWNSYS command displays a confirmation panel. If your system is at a release level earlier than V6R1, use the following command to cause a confirmation of the power down system command to be the default:

  ```
  ADDENVVAR ENVVAR(QIBM_PWRDWNSYS_CONFIRM) VALUE(*INTERACT) LEVEL(*SYS)
  ```

- If you are experiencing abnormal IPLs, you might need to increase the QPWRDWNLMT system value, which sets the amount of time that is allowed for jobs to end during shut down. Use the following command:

  ```
  CHGSYSVAL SYSVAL(QPWRDWNLMT) VALUE(1000)
  ```

  The CPI091D message (previously the ending abnormal, reason code 7) is sent to the history log if a shut down has timed out.

Look at the setting for the QENDJOBLMT system value whenever you change the setting for the QPWRDWNLMT system value. QPWRDWNLMT must be larger than QENDJOBLMT. The difference between these two values must be large enough to allow the final cleanup steps to occur.

- If you have to use the ENDJOBABN command regularly on the same job, obtaining the call stack of the job before you use the ENDJOBABN can be helpful in investigating why the job does not end.
- User trace information can be left after problems are encountered and debugged. If the WRKOBJ OBJ(QUSRSYS/QP0Z*) OBJTYPE(*USRSPC) command displays a large number of objects, you can use the DLTUSRTRC JOB(*ALL) command to delete them.
- Connecting to the system using virtual devices works best when the connection request supplies the name of a device description. This method simplifies the device selection logic and avoids contention problems that can occur when large numbers of users connect at the same time.

  You can configure the display device name on the "Session Parameters" for PCOMM or PC5250 sessions. With the Telnet command, use the RMTVRTDSP parameter to specify the device name.
- Customized sign-on prompts tend to add value, particularly when an operator or user might connect to several different systems.
- Supporting multiple time zones is most often done with multiple partitions. Make sure that all time and date information is associated clearly with a time zone (or UTC). Do not underestimate the complexity of time and date calculations that involve daylight savings time transitions. A user interface is best done in local time, but daylight savings time produces some ambiguous values and some invalid values. Store UTC. Calculate in UTC. Learn to think in UTC.
- If you run in a partition with uncapped processors, expect to see CPU percentages that exceed 100%. Even without uncapped processors, you get some sampling error that can drive the measurements above 100%, although this error is mostly an issue with small time interval samples, which is *not* a best practice.
- When using the QCTLSBSD system value, keep it simple. The controlling subsystem needs to handle only the console and the startup program. Use QCTL rather than QBASE.
- Set the HSTLOGSIZ system value to *DAILY.
- For the QLIBLCKLVL system value, use 0 to minimize system lock table use.

Summary of the system value settings

Table 4 summarizes the system values that we discuss in this paper and includes the IBM recommendation for the setting versus the shipped default.

Table 4 System values: Recommended versus shipped values

System Value	Recommendation	Shipped Setting
QSPLFACN	*DETACH	*KEEP
QMAXSPLF	9999	9999
QRCLSPLSTG	Number of days between normal peek usage + 1	8
QMAXJOB	Normal Maximum number of jobs/ 0.90	163520
QJOBMSGQFL	*WRAP	*NOWRAP
QJOBMSGQMX	8	16
QLOGOUTPUT	*JOBLOGSVR	*JOBEND
QDEVRCYACN	*ENDJOBNOLIST	*DSCMSG
QSTGLOWACN	*ENDSYS	*MSG
QPFRADJ	3	2
QHSTLOGSIZ	*DAILY	5000
QCTLSBSD	QSYS/QCTL	QSYS/QBASE
QLIBLCKLVL	0	1

SMBJOB exit program

Run-away usage of the SMBJOB command can consume large numbers of job table entries. You can register and run a SBMJOB command exit program to avoid loops that use up job table entries. Example 2 provides sample code of the exit program. We provide this code "as is."

Example 2 Sample SMBJOB exit program

```
/********************************************************************/
/*                                                                  */
/* ZSBMEXIT - Exit program for SBMJOB command                       */
/*     Registered under QIBM_QCA_RTV_COMMAND exit point.            */
/*                                                                  */
/* Purpose: Restrict use of the Submit Job (SBMJOB) command         */
/*     to avoid excessive system demand. This is primarily designed */
/*     to detect and stop a looping job that is issuing SBMJOB      */
/*     commands.                                                    */
/*                                                                  */
/*     This is a sample program that demonstrates the concept of    */
/*     implementing a policy for the use of the SBMJOB command.     */
/*     To implement your own policy, you need to adjust time values */
/*     and adjust the number of commands that you allow.            */
/*     Refer to the i5/OS Information Center for license and        */
/*     disclaimer information about sample programs.                */
/*                                                                  */
/* Note:                                                            */
/*     You must create the program such that all users are          */
/*     authorized to find and call the program.  Otherwise,         */
/*     you get CPI0001 "Exit program not called. Reason code &1."   */
/*     and no limit is enforced for those users.                    */
/*     In some environments, it may be necessary to use *OWNER      */
/*     for the User profile (USRPRF) parameter of the               */
/*     Create CL Program (CRTCLPGM) command.                        */
/*                                                                  */
/* Registration (sample):                                           */
/*     ADDEXITPGM EXITPNT(QIBM_QCA_RTV_COMMAND) FORMAT(RTVC0100) +   */
/*         PGMNBR(*HIGH) PGM(lib-name/ZSBMEXIT) +                   */
/*         THDSAFE(*YES) MLTTHDACN(*RUN) +                          */
/*         PGMDTA(*JOB 20 'SBMJOB    QSYS      ') +                 */
/*         TEXT('Limit SBMJOB repetitions')                         */
/*                                                                  */
/*   You can use the Work with Registration Info (WRKREGINF)        */
/*   command, '8=Work with exit programs', and '4=Remove' to remove */
/*   this registration.                                             */
/*                                                                  */
/* Implementation notes:                                            */
/*     This code keeps a data area (QTEMP/ZSBMEXIT) with the        */
/*     following information:                                       */
/*                                                                  */
/*     Start  Length  Description                                   */
/*       1      52      Descriptive eyecatcher                      */
/*       53     4       Count of SBMJOB commands done by this job.  */
/*       57     8       Timestamp of beginning of interval          */
/*       65     8       Timestamp of most recent SBMJOB             */
/*                                                                 */
/*     If a single job tries to issue more than 100 SBMJOB commands, */
/*     this exit program will stop that job.  The count is reset     */
/*     if there is a significant gap (&TIMEGAP, 5 minutes) or if     */
/*     the overall rate is less than &SBMLIMIT (100) SBMJOB commands */
/*     in &TIMELMT (3 hours).  By working in UTC and using           */
/*     two different mechanisms to protect long-running jobs from    */
/*     exceeding the threshold, this program is less sensitive to    */
/*     clock changes that would produce a negative time interval.    */
/*                                                                 */
/* CHANGE ACTIVITY:                                                */
/*   Version    Date         Description                           */
/*   ---------  -----------  ------------------------------------- */
/*   1.0        2007-Oct-09  Prototype and demonstration           */
/*                                                                 */
/*******************************************************************/
PGM PARM(&INPARM)

/*******************************************************************/
/* Declares for input parameters.                                  */
/* The SBMJOB command string is available, but not used by this    */
/* program.                                                        */
/*******************************************************************/
  DCL VAR(&INPARM)   TYPE(*CHAR) LEN(32767)

/*******************************************************************/
/* Constants that define the policy being enforced.                */
/* You can use a hex calculator to compute different time values   */
/* and convert the first four bytes into a (base 10) integer.      */
/* Close is good enough. It is possible to get the Convert Date and */
/* Time Format (QWCCVTDT) API to give you timestamps that you can  */
/* use to produce an exact answer, but this program only looks at  */
/* the first four bytes of the timestamps.                         */
/*******************************************************************/
  DCL VAR(&SBMLIMIT) TYPE(*DEC)  LEN(15 0) VALUE(100) /* +
         The maximum number of SBMJOB commands allowed within +
         a time interval. Adjust &CHSBMLMT, if necessary. */
  DCL VAR(&CHSBMLMT) TYPE(*CHAR) LEN(3) /* Character view +
         of &SBMLIMIT.  Adjust LEN() if &SBMLIMIT > 999 */
  DCL VAR(&TIMEGAP)  TYPE(*DEC)  LEN(15 0) VALUE(286) /* +
         Comparison value for 5 minute interval. +
         Full 8-byte TOD form is X'0000011E1A300000'. */
  DCL VAR(&TIMELMT)  TYPE(*DEC)  LEN(15 0) VALUE(10299) /* +
         Comparison value for 3 hour interval. +
         Full 8-byte TOD form is X'0000283BAEC00000'. */

/*******************************************************************/
/* Variables used to access the information in the data area.      */
/*******************************************************************/
  DCL VAR(&DTAA)     TYPE(*CHAR) LEN(2000) /* Copy of data area +
         QTEMP/ZSBMEXIT */
  DCL VAR(&INITCNT)  TYPE(*CHAR) LEN(4) VALUE(X'00000001')
  DCL VAR(&CHARCNT)  TYPE(*CHAR) LEN(4) /* *CHAR view of count */
  DCL VAR(&DECCNT)   TYPE(*DEC)  LEN(15 0) /* *DEC view of count */
  DCL VAR(&STRTOD)   TYPE(*CHAR) LEN(8) /* Starting time */
  DCL VAR(&LSTTOD)   TYPE(*CHAR) LEN(8) /* Last SBMJOB time */

/*******************************************************************/
/* Identification for the current job.  This information can be used */
/* to provide different limits for different jobs.                 */
/*******************************************************************/
  DCL VAR(&CJOBNAME) TYPE(*CHAR) LEN(10) /* Job name of +
         the job that this command exit is running in. */
  DCL VAR(&CJOBUSER) TYPE(*CHAR) LEN(10) /* Job user name of +
         the job that this command exit is running in. */
  DCL VAR(&CJOBNBR)  TYPE(*CHAR) LEN(6) /* Job number of +
         the job that this command exit is running in. */
  DCL VAR(&CURUSER)  TYPE(*CHAR) LEN(10) /* Current user name of +
         the job that this command exit is running in. */
  DCL VAR(&CJOBTYPE) TYPE(*CHAR) LEN(1) /* Job type of +
         the job that this command exit is running in. */

/*******************************************************************/
/* Interface to the Convert Date and Time Format (QWCCVTDT) API      */
/*******************************************************************/
  DCL VAR(&INFMT)    TYPE(*CHAR) LEN(10) VALUE('*CURRENT  ')
  DCL VAR(&INTOD)    TYPE(*CHAR) LEN(8)  VALUE(X'0000000000000000')
  DCL VAR(&OUTFMT)   TYPE(*CHAR) LEN(10) VALUE('*DTS      ')
  DCL VAR(&CURTOD)   TYPE(*CHAR) LEN(8)  VALUE(X'0000000000000000')
  DCL VAR(&ERRCOD)   TYPE(*CHAR) LEN(8)  VALUE(X'0000000000000000')
  DCL VAR(&INTZ)     TYPE(*CHAR) LEN(10) VALUE('*UTC      ')
  DCL VAR(&OUTTZ)    TYPE(*CHAR) LEN(10) VALUE('*UTC      ')
  DCL VAR(&TZINFO)   TYPE(*CHAR) LEN(1)
  DCL VAR(&LENTZI)   TYPE(*CHAR) LEN(4)  VALUE(X'00000000')
  DCL VAR(&PRECIND)  TYPE(*CHAR) LEN(1)  VALUE('0')

/*******************************************************************/
/* Miscellaneous variables.                                        */
/*******************************************************************/
  DCL VAR(&CANRESET) TYPE(*CHAR) LEN(1) VALUE('N') /* Allowed to +
         reset the times and count */
  DCL VAR(&TIME1)    TYPE(*DEC)  LEN(15 0) /* For time calculation */
  DCL VAR(&TIME2)    TYPE(*DEC)  LEN(15 0) /* For time calculation */
  DCL VAR(&INTERVL)  TYPE(*DEC)  LEN(15 0) /* For time calculation */

  DCL VAR(&MSGKEY)   TYPE(*CHAR) LEN(4) /* Message key */
  DCL VAR(&OPRRPLY)  TYPE(*CHAR) LEN(1) /* Reply message */

/*******************************************************************/
/* Program-scoped error handling.                                  */
/*******************************************************************/
  MONMSG MSGID(CPF0000)  EXEC(GOTO CMDLBL(ENDOFPGM))

/*******************************************************************/
/* Find out where this exit program is running.                    */
/* For some cases, this program should not make any decisions.      */
/*******************************************************************/

  QSYS/RTVJOBA JOB(&CJOBNAME) USER(&CJOBUSER) NBR(&CJOBNBR)  +
         CURUSER(&CURUSER) TYPE(&CJOBTYPE) /* Retrieve info +
         about the current job */

  IF (   (&CJOBTYPE *EQ 'X') /* SCPF */ +
     *OR (&CJOBTYPE *EQ 'S') /* System job */ +
     *OR (&CJOBTYPE *EQ 'M') /* Subsystem monitor job */ +
     ) THEN(GOTO CMDLBL(ENDOFPGM)) /* Do not operate if invoked +
         in system or subsystem jobs. */

/*********************************************************************/
/* When first invoked, create and initialize the data area.          */
/* If two threads try this at the same time, one thread will receive */
/* an error and will continue by updating the data area.             */
/*********************************************************************/

  QSYS/CHKOBJ OBJ(QTEMP/ZSBMEXIT) OBJTYPE(*DTAARA)
  MONMSG MSGID(CPF0000) EXEC(DO)

    CHGVAR VAR(%SST(&DTAA 1 52)) VALUE('ZSBMEXIT DATA, +
         QIBM_QCA_RTV_COMMAND FOR SBMJOB      ') /* Eyecatcher */
    CHGVAR VAR(%SST(&DTAA 53 4)) VALUE(&INITCNT) /* Initial count +
         is 1. */

    QSYS/CALL PGM(QSYS/QWCCVTDT) +
         PARM(&INFMT  &INTOD  &OUTFMT &CURTOD &ERRCOD +
              &INTZ &OUTTZ &TZINFO &LENTZI &PRECIND) /* Get +
           current time in UTC. */

    CHGVAR VAR(%SST(&DTAA 57 8)) VALUE(&CURTOD) /* Initial +
         start-of-interval. */
    CHGVAR VAR(%SST(&DTAA 65 8)) VALUE(&CURTOD) /* Initial +
         time of last SBMJOB. */

    QSYS/CRTDTAARA DTAARA(QTEMP/ZSBMEXIT) TYPE(*CHAR) LEN(2000) +
        VALUE(&DTAA) /* Create and initialize */
    MONMSG MSGID(CPF0000) EXEC(GOTO CMDLBL(UPDDTAA))

    GOTO CMDLBL(ENDOFPGM) /* Finished, only one SBMJOB done. */
  ENDDO

/*********************************************************************/
/* If the data area already exists, prepare to update the data area. */
/*********************************************************************/

UPDDTAA:
  QSYS/ALCOBJ OBJ((QTEMP/ZSBMEXIT *DTAARA *EXCLRD)) +
      WAIT(3600) SCOPE(*THREAD) /* Lock the data area */
  MONMSG MSGID(CPF1002) EXEC(GOTO CMDLBL(UPDDTAA)) /* For timeout, +
        try again. This may be needed to stop a thread from doing +
        additional SBMJOB commands when another thread has already +
        detected a problem. */

  QSYS/CALL PGM(QSYS/QWCCVTDT) +
      PARM(&INFMT  &INTOD  &OUTFMT &CURTOD &ERRCOD +
          &INTZ &OUTTZ &TZINFO &LENTZI &PRECIND) /* Get +
        current time in UTC. This is required to be greater than +
        the values stored in the data area, so this time stamp +
        must be obtained AFTER locking the data area. */
  MONMSG MSGID(CPF0000) EXEC(GOTO CMDLBL(UNLKDTAA))

  QSYS/RTVDTAARA DTAARA(QTEMP/ZSBMEXIT *ALL) RTNVAR(&DTAA)
  MONMSG MSGID(CPF0000) EXEC(GOTO CMDLBL(UNLKDTAA))

  CHGVAR VAR(&CHARCNT) VALUE(%SST(&DTAA 53 4)) /* *CHAR count */
  CHGVAR VAR(&DECCNT) VALUE(%BIN(&CHARCNT 1 4)) /* *DEC count */
  CHGVAR VAR(&STRTOD) VALUE(%SST(&DTAA 57 8)) /* Starting time */
  CHGVAR VAR(&LSTTOD) VALUE(%SST(&DTAA 65 8)) /* Last SBMJOB time */

/*******************************************************************/
/* One way to handle the question of controlled-rate is to look for  */
/* a gap that is large enough to be significant.                    */
/*******************************************************************/
  IF COND(&CURTOD *GE &LSTTOD) THEN(DO) /* Alphabetic compare +
         to ensure a positive time interval. */
    CHGVAR VAR(&TIME1) VALUE(%BIN(&LSTTOD 1 4)) /* Use first half +
         of 8-byte timestamp */
    CHGVAR VAR(&TIME2) VALUE(%BIN(&CURTOD 1 4)) /* Use first half +
         of 8-byte timestamp */
    CHGVAR VAR(&INTERVL) VALUE(&TIME2 - &TIME1) /* Find time +
         interval since last SBMJOB */
    IF COND(&INTERVL *GT &TIMEGAP) +
      THEN(CHGVAR VAR(&CANRESET) VALUE('Y')) /* Look for +
         a 5 minute minute gap with no SBMJOB.  If found, +
         job is probably not doing an excessive number of +
         SBMJOB commands, so set flag to reset the count. */
  ENDDO

/*******************************************************************/
/* Another way to handle the question of controlled-rate is to check */
/* for a threshold number over a significant interval. In this case, */
/* we do not check the number of SBMJOB commands (just the interval) */
/* because exceeding the threshold is handled at the time we pass    */
/* the threshold.                                                    */
/*******************************************************************/
  IF COND(&CURTOD *GE &STRTOD) THEN(DO) /* Alphabetic compare +
         to ensure a positive time interval. */
    CHGVAR VAR(&TIME1) VALUE(%BIN(&STRTOD 1 4)) /* Use first half +
         of 8-byte timestamp */
    CHGVAR VAR(&TIME2) VALUE(%BIN(&CURTOD 1 4)) /* Use first half +
         of 8-byte timestamp */
    CHGVAR VAR(&INTERVL) VALUE(&TIME2 - &TIME1) /* Find time +
         interval for all counted SBMJOB commands */
    IF COND(&INTERVL *GT &TIMELMT) +
      THEN(CHGVAR VAR(&CANRESET) VALUE('Y')) /* Look for +
         a 3 hour sample size (that did not cause trouble). +
         If found, job is probably not doing an excessive  +
         number of SBMJOB commands, so set flag to reset the count. */
  ENDDO

/*******************************************************************/
/* If this job does not appear to be doing too many SBMJOB commands, */
```

```
/* we can reset the stored information so that long-running jobs     */
/* do not exceed the threshold that would trigger corrective action. */
/*********************************************************************/
  IF COND(&CANRESET *EQ 'Y') THEN(DO)
    CHGVAR VAR(%SST(&DTAA 53 4)) VALUE(&INITCNT) /* Reset count +
         to 1. */
    CHGVAR VAR(%SST(&DTAA 57 8)) VALUE(&CURTOD) /* Reset +
         start-of-interval. */
    CHGVAR VAR(%SST(&DTAA 65 8)) VALUE(&CURTOD) /* Reset +
         time of last SBMJOB. */
  ENDDO
/*********************************************************************/
/* If we do not reset the count, we will increment the count.        */
/*********************************************************************/
  ELSE CMD(DO)
    CHGVAR VAR(&DECCNT) VALUE(&DECCNT + 1) /* Increment count */
    CHGVAR VAR(%BIN(&CHARCNT 1 4)) VALUE(&DECCNT) /* Convert to +
         *CHAR view */
    CHGVAR VAR(%SST(&DTAA 53 4)) VALUE(&CHARCNT) /* Update count +
         in copy of data area. */
    CHGVAR VAR(%SST(&DTAA 65 8)) VALUE(&CURTOD) /* Reset +
         time of last SBMJOB. */
/*********************************************************************/
/* Check the count of SBMJOB commands against a threshold.           */
/* Send inquiry message to operator, wait for a response.            */
/* Need v5r4 or better to do ENDJOB JOB(*) OPTION(*IMMED)            */
/* Probably best to DLYJOB DLY(99999) and allow dumps/etc.           */
/* Could allow additional responses - for example, allow an          */
/* unlimited number of additional SBMJOB commands from this job.     */
/*********************************************************************/
    IF COND(&DECCNT *GT &SBMLIMIT) THEN(DO)

      CHGVAR VAR(&CHSBMLMT) VALUE(&SBMLIMIT) /* Convert the +
         limit to a character string. */
      MONMSG MSGID(CPF0000) EXEC(DO)
        CHGVAR VAR(&CHSBMLMT) VALUE('NNNN') /* Use 'NNNN' if +
         value was too large to convert. */
        MONMSG MSGID(CPF0000)
      ENDDO

      QSYS/SNDPGMMSG MSG('Too many SBMJOB commands. Job' +
         *BCAT &CJOBNBR *CAT '/' *CAT &CJOBUSER *TCAT '/' +
         *CAT &CJOBNAME *BCAT 'appears to be looping.' +
         *BCAT 'Use the ENDJOB command to end the looping' +
         *BCAT 'job or reply ''I'' to allow an additional' +
         *BCAT &CHSBMLMT *BCAT 'SBMJOB commands.' +
         *BCAT 'All other replies result in DLYJOB.') +
         TOMSGQ(*SYSOPR) MSGTYPE(*INQ) KEYVAR(&MSGKEY) /* Send +
         an inquiry to QSYSOPR so that the operator can override +
         the decision made by this exit program. */
      MONMSG MSGID(CPF0000) EXEC(DLYJOB DLY(99998))
      QSYS/RCVMSG MSGQ(*PGMQ) MSGTYPE(*RPY) MSGKEY(&MSGKEY) +
          MSG(&OPRRPLY) WAIT(*MAX) RMV(*NO) /* Receive a reply. */
      MONMSG MSGID(CPF0000) EXEC(DLYJOB DLY(99997))
      IF COND(   (&OPRRPLY *EQ 'I') +
             *OR (&OPRRPLY *EQ 'i') ) THEN(DO) /* See if +
         operator wishes to permit the job to continue (for +
         another &SBMLIMIT number of jobs. */
        CHGVAR VAR(%SST(&DTAA 53 4)) VALUE(&INITCNT) /* Reset count +
             to 1. */
        CHGVAR VAR(%SST(&DTAA 57 8)) VALUE(&CURTOD) /* Reset +
             start-of-interval. */
        CHGVAR VAR(%SST(&DTAA 65 8)) VALUE(&CURTOD) /* Reset +
             time of last SBMJOB. */
      ENDDO
      ELSE CMD(DO)
        DLYJOB DLY(99999) /* Stop this thread while holding the +
         exclusive lock on the data area (thus stopping any +
         other threads that try to do SBMJOB. */
      ENDDO

    ENDDO
  ENDDO

/**********************************************************************/
/* Update the data area and unlock it.                                */
/**********************************************************************/
  QSYS/CHGDTAARA DTAARA(QTEMP/ZSBMEXIT *ALL) VALUE(&DTAA) /* Update +
      the data in the data area. */
    MONMSG MSGID(CPF0000) /* On error, continue with unlock */
UNLKDTAA:
  QSYS/DLCOBJ OBJ((QTEMP/ZSBMEXIT *DTAARA *EXCLRD)) +
      SCOPE(*THREAD) /* Unlock the data area */

/**********************************************************************/
/* End of program                                                     */
/**********************************************************************/
ENDOFPGM:

ENDPGM
```

The team that wrote this paper

This paper was produced by a team of specialists from around the world working at the International Technical Support Organization (ITSO), Rochester Center.

Dawn May is a Senior Technical Staff Member at IBM in Rochester, Minnesota. Dawn's current position is the technical lead for the IBM i Systems Software Development organization and IBM i Business Architect. In this position, she has the responsibility for the overall strategy and plan for the core components of IBM i. Dawn also has technical leadership responsibilities for IBM i performance and diagnostic tools. She led the development of the 6.1 performance tools enhancements, which include the new Performance tasks in the IBM Systems Director Navigator for i5/OS® Web console. Past responsibilities included testing, SNA and TCP/ IP communications development, service tools development, and work management development.

Angela Newton is an advisory software engineer on the Work Management team at IBM in Rochester, Minnesota. She has worked in the lab for the past 19 years on the IBM i operating system, iSeries®, and AS/400®.

Mike Rusell is a software developer on the work management team for IBM i. He previously worked in operating system development on OS/2® for PowerPC®, AS/400, 9370, System/38™, and System/3.

Dan Tarara is an advisory software engineer in the IBM Rochester Programming Laboratory. He is currently in the Work Management area of the IBM i. He has been working for 32 years in operating systems and microcode. Dan has a Bachelor of Arts degree from Saint Mary's University of Minnesota.

Kevin Vette is an Advisory Software Engineer at IBM Rochester. He has worked at IBM in Rochester for 34 years on System/3, System/38, System 36, and IBM i. He currently works on the spool component of IBM i. He is a graduate of Concordia College, Moorhead, Minnesota.

Thanks to the following people for their contributions to this project:

Cheryl Gera
Hernando Bedoya
International Technical Support Organization, Rochester Center

Bob Baron
Terry Luebbe
IBM Systems and Technology Group, System Software Development, Rochester, Minnesota

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

This document REDP-4454-00 was created or updated on September 17, 2008.

Send us your comments in one of the following ways:
- Use the online **Contact us** review Redbooks form found at:
  `ibm.com/redbooks`
- Send your comments in an e-mail to:
  `redbooks@us.ibm.com`
- Mail your comments to:
  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400 U.S.A.

Redpaper™

Trademarks