IBM

eserver

iSeries

Work management

IBM

eserver

iSeries

Work management

Contents

Work management

Work management is an important building block within the iSeries server operating system. Its functions are the foundation through which all work enters the system, is processed, run, and completed on iSeries servers. Whether you run a simple batch job once a week or you call an application daily (like Lotus Notes), work management helps manage the jobs and objects that run on your system. It also supports the commands and internal functions necessary to control system operations and allocate resources to applications when needed.

The iSeries server is set up and ready to use. Most users will not need to change the default settings. However, if you need to tailor the work management piece to fit your company, you will need to understand the terms and concepts associated with it and how they integrate with each other to provide you with the best performance from your iSeries server.

Whether you are a experienced iSeries user or just learning, this topic gives you an easy-to-understand view of work management. This topic contains different entry points, so you choose where you want to start learning about work management.

A job's life
Follow a job through its life cycle in the work management infrastructure—use our interactive graphic to click your way to more detailed information about work management .

Manage daily work
Find out the daily tasks you can perform to efficiently manage work from iSeries Navigator and when to perform these tasks. From checking job logs to monitoring system activity, you will learn important daily tasks involved with work management.

The structure of your system
Learn the terms and concepts associated with work management(including job, job queues, subsystems, and memory pools) that you can use to manage work on an iSeries server.

How work gets done
Find out what you will need to do to get work done on your iSeries server. Set up job queues, allocate memory to your subsystems and understand what happens to the job after it finishes running.

Troubleshoot work management
Read about how to resolve the problems with jobs through iSeries Navigator.

See the What's new topic for the new and changed information and see the Print this topic if you want to print the PDF for this entire topic.

Related Information

IBM manuals contain technical information, know-how, and "how-to" information.

What's new for V5R2

In V5R2, many new functions have been added to the work management component in iSeries Navigator. These new features and functions are integrated into the work management structure, so you can still choose where you want to start learning about the work management component: A job's life (interactive graphic), manage daily work, iSeries server structure, and how work gets done. Each of these areas represent a different level of understanding of work management. Whether you are an experienced iSeries user or just learning, these articles give you an easy-to-understand view of work management.

New iSeries Navigator GUI function

Many of the work management functions and tasks that users and administrators were able to complete through the character-based interface now can be done through iSeries Navigator. Below is a list of the new functions.

System status

- This dialog is accessible from both the system connection and from the work management folder in iSeries Navigator.
- This dialog provides a single location from which the user can identify and potentially resolve problems as well as access various iSeries Navigator functions, such as active jobs, logical partitions, memory pools, and disk pools.

Jobs

- Added the following job list windows:

  Jobs running in a subsystem
  Jobs for a transaction
  Jobs using an integrated file system(IFS) object
  Jobs using a tape device
- Identify the program or procedure that issues a lock request
- Work with locked members for a specific locked object
- Work with locked rows for a specific locked member
- Work with jobs and lock spaces that have locks on an object, member, or row
- Thread management:

  View threads running under a specific job
  End threads
  View thread properties, including Elapsed Performance Statistics
  Change the run priority of a thread
  Work with the call stack for a thread
  Work with the library list for a thread
  Work with locks for a thread
  Work with transactions attached to a thread
- Additional job actions:

  Work with transactions attached to a specific active job
  Work with last SQL statement that was run by a specific active job
  Date and time stamp added to the elapsed performance statistics window
  Work with locked objects for an active job, thread, transaction, or lock space
- Additional job properties:

  Detach printer output option
  New disk pool group property on the Other page of job property sheets
  Detailed status values for when a job is waiting on a lock, waiting on a dequeue, or waiting on a lock space that identify the item being waited on
  Detailed status value that indicates that a job in a common job list no longer exists on the system
  Launch Printer Output, Job Log, and Threads from properties pages

Job Queues

- Move jobs to the top of other job queues
- Clear a job queue without creating a job log

Output Queues
- View printer output on output queues
- Move printer output within and between output queues
- Changed spooled file to printer output file

How to see what's new or changed

To help you see where technical changes have been made, this information uses:
- The

  »

  image to mark where new or changed information begins.
- The

  «

  image to mark where new or changed information ends.

To find other information about what's new or changed this release, see the Memo to Users

.

Print this topic

You can view or download a PDF version of these documents for viewing or printing. You must have Adobe(R) Acrobat(R) Reader installed to view PDF files. You can download a copy from Adobe

.

To view or download the PDF version, select the following:
- Work Management (about 173 KB or 40 pages)
- System Values (about 2430 KB or 277 pages)

Other information

You can also view or print the V4R5 Work Management manual PDF:
- V4R5 Work Management

  (about 2720 KB or 573 pages)

To save a PDF on your workstation for viewing or printing:
1. Open the PDF in your browser (click the link above).
2. In the menu of your browser, click **File**.
3. Click **Save As...**
4. Navigate to the directory in which you would like to save the PDF.
5. Click **Save**.

Manage daily work

As a system operator or administrator, one of your tasks is to keep your server running smoothly. This means you monitor, manage, and ensure that your jobs, job queues, subsystems, memory pools, job logs, and output queues function properly.

The topics in this section give you information on the different types of daily work management tasks as well as other tasks you might need to perform on your iSeries server. Each subtopic explains why it is important to do these tasks, as well as how to complete them.

Monitor system activity
Monitoring your system is an important daily activity. You can accomplish this in a variety of ways, such as using iSeries Navigator and iSeries Navigator Management Central. The tasks in these subtopics follow:

- Work with system status
- Monitor system performance
- Work with monitors

Manage jobs and threads
Whether you are asked to report the status of a particular job or thread or to monitor a job or thread's performance, you can easily find most of the answers you need in iSeries Navigator. The tasks in these subtopics follow:

- Find a job on the iSeries server
- Determine the status of a job
- View performance statistics for a job
- End a job
- Actions done to a job
- View threads running under a specific job
- View thread properties
- End a thread

Manage job queues
Job queues are an important element in the life cycle of a batch job. Job queues help control the rate at which batch jobs enter a subsystem. The tasks in these subtopics follow:

- View jobs on the job queue
- Change the priority of a job within a job queue
- Move jobs to different job queues

Manage subsystems
Because jobs run in subsystems, you may need to monitor subsystem activity for potential problems that could affect a job's ability to run. The tasks in these subtopics follow:

- Monitor a subsystem
- View jobs in a subsystem
- Start a subsystem
- End a subsystem

Manage memory pools
Memory pools allocate memory to subsystems so that jobs can run. It is important that when jobs run they get enough memory to complete efficiently. The tasks in these subtopics follow:

- Monitor the number of jobs in a memory pool
- Monitor the number of subsystems in a memory pool

- Check memory use
- Change the size of a memory pool

Manage job logs
Job logs contain information related to requests entered for a job, such as commands in the job, commands in the program, and messages. The tasks in these subtopics follow:

- Access job logs for active jobs, including server jobs
- Access printer output

Manage output queues
Output queues help you manage printer output created when a job ends. It is important to understand how to effectively maintain your output queues so that your printed output processes smoothly. The tasks in these subtopics follow:

- View output queues on the system
- Clear output queues
- Move output between and within output queues

Monitor system activity

Monitoring system activity is one of the many important tasks in the day of an administrator. Monitoring the flow of work through the system is only a piece of the information that should be monitored on a daily basis. IBM offers a variety of tools to help you monitor your system performance from basic system checking using system status to advanced system monitoring with Management Central.

Work with system status
In iSeries Navigator, the system status window gives you the ability to view and access various system functions on a system in one convenient location.

Monitor system performance
The Management Central function in iSeries Navigator has system monitors that collect and display real-time performance data from which you can track and troubleshoot system performance problems.

Work with monitors
Monitor your jobs and servers, your message queues, changes to selected files, and business-to-business transaction activity.

Work with system status

Modeled after the top half of the Work with System Status (WRKSYSSTS) display in the character-based interface, the System Status dialog offers a quick and easy way to check the status of a system. Management Central allows you to monitor more indepth functions through the use of system monitors.

The different functions that you can do from the system status window are:

- View CPU usage
- View the total number of jobs, active jobs, and the maximum number of jobs allowed on the system
- View the number of active threads on the system
- View the percentage of addresses (permanent and temporary) used on the system
- View the total disk space
- View the system disk pool capacity and usage
- View the number of processors on your system

Note: Three different **Processors** pages exist depending on the type of iSeries system you have. You may see additional processor related information depending on the configuration of your system:

System with no partitions
System with partition, dedicated processors
System with partition, shared processors

For more information on logical partitioning on the iSeries system, see Logical partitions.

– View the total memory on the system
– View the temporary storage used
– View the current amount of temporary storage used and the maximum amount used since the last system restart
– Access active jobs
– Access jobs and storage system values
– Access disk pools
– Access active memory pools
– Access the Configure Logical Partitions dialog

You can access the System Status dialog from the **System** folder or the **Work Management** folder within iSeries Navigator.

To get to system status from the **System** folder:
1. In iSeries Navigator, expand **My Connections**.
2. Right-click the connection on which you want to work and select **System Status**.

For more information on the different tasks that you can complete using system status, see the iSeries Navigator help.

Managing jobs and threads

Since work done on your system is in the form of jobs and threads, it is important that you can find, track, and manage them within your system.

These subtopics explain how to find a particular job, how to determine the status of a job, how to monitor the performance of a job, how to end a job, what actions you can perform on a job, how to view threads and their properties, and how to end threads.
- Find a job on the iSeries server
- Determine the status of a job
- View performance statistics for a job
- End a job
- Job actions
- View threads running under a specific job
- View thread properties
- End a thread

For more information on the different tasks you can perform on jobs and threads, see the iSeries Navigator help.

For more detailed information on jobs and the types of jobs on an iSeries server, see Jobs. For more detailed information on threads, see Threads.

Find a job on the iSeries server

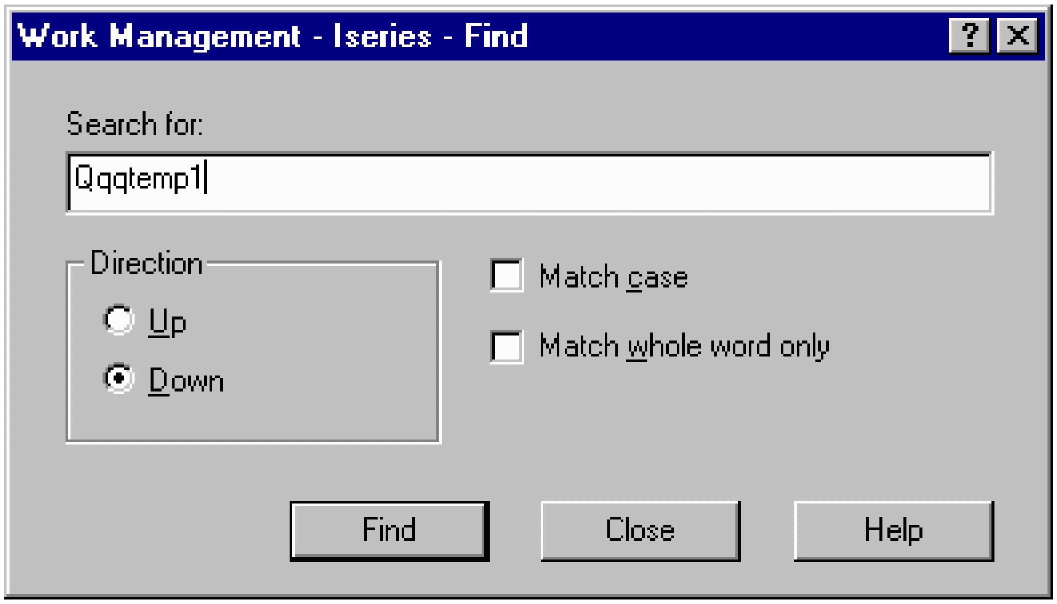
It is important to understand how to find jobs on your iSeries server. Whatever the reason, at some point in time you may need certain information from a particular job. In iSeries Navigator, you can do a **Find** on all your jobs or you can narrow your search using the **Include...** function followed by Find. The Include... function allows you to put limitations on what is displayed in iSeries Navigator. For example, instead of doing a Find on hundreds of jobs, you can run an Include... to display only certain job types. Or, you can display only those jobs with specific job user IDs.

From a performance standpoint, if you have lots of jobs on the system, it is recommended that you use the Include... function to narrow the number of jobs searched. If you have a lot of jobs on the system, searching through all of them can hinder system performance.

Note: You can use the menu bar **Find** and **Include...** throughout work management where you find jobs. You can also use these tools to find job queues, subsystems, and memory pools in the same manner. Remember that you need to click on the area you want to search before you can use these tools.

To find a job using the **Find (Ctrl+F)** option, do the following:

1. In iSeries Navigator, expand **My Connections.**
2. Expand the connection for your **iSeries server**.
3. Expand **Work Management.**
4. Click **Active Jobs**; then select **Edit**.
5. From the Edit menu, select **Find (Ctrl+F)**.
6. In the **Search for** field, type the job ID you want to find (for example, Qqqtemp1). All the job columns are searched for your job.


Work Management - Iseries - Find

Search for: Qqqtemp1

Direction: Up [✓ Down]

Match case: not checked

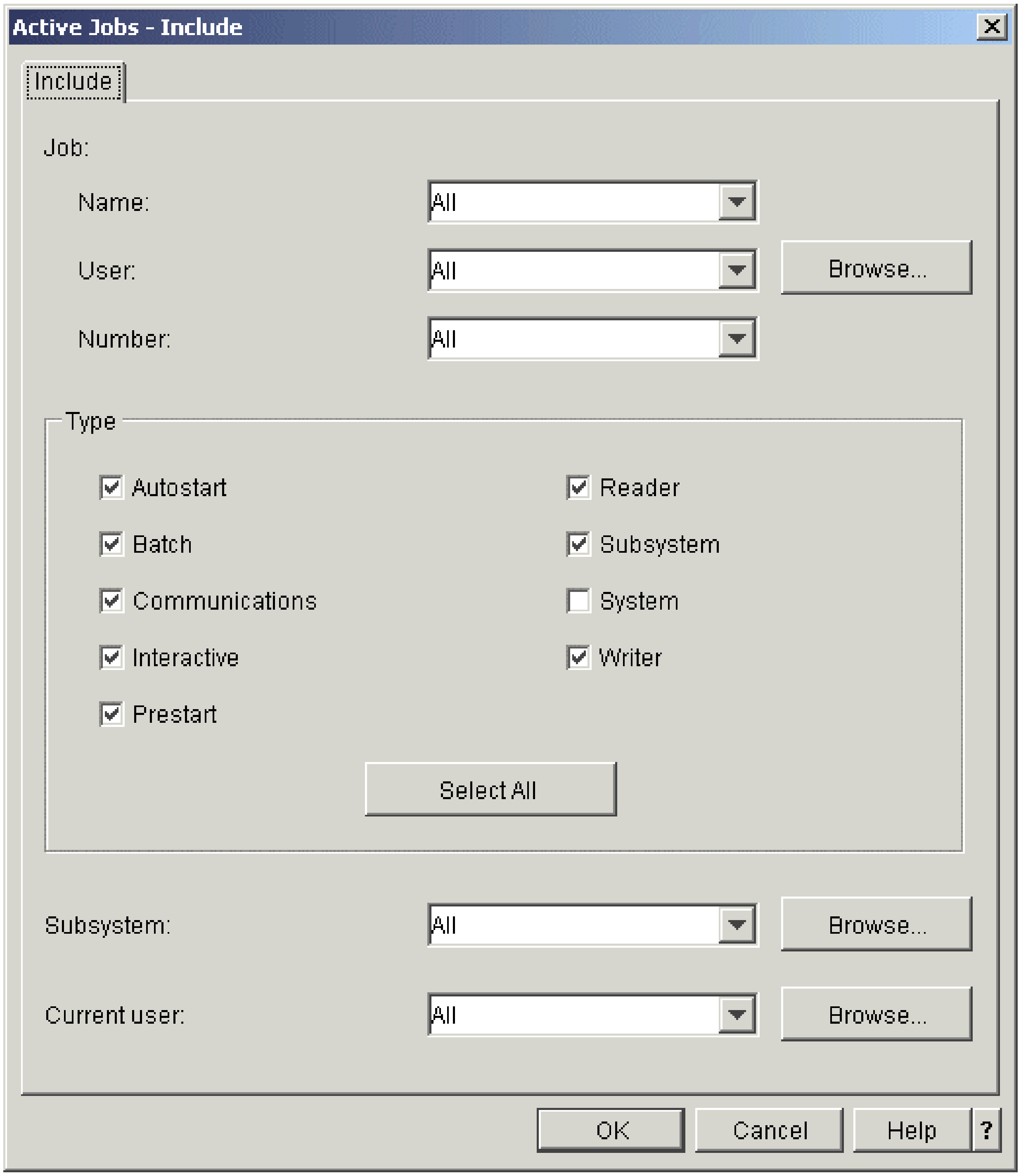
Match whole word only: not checked

7. Click **Find.** iSeries Navigator will highlight your job once it is found.
   Note: Remember that job names are only case sensitive when enclosed in quotations (for example, ″MyJob″). If the job name is not enclosed in quotations, then it is not case sensitive.

To limit the information that is displayed using **Include...** function; do the following:

1. In iSeries Navigator, expand **My Connections.**
2. Expand the connection for your **iSeries server**.

3. Expand **Work Management**.
4. Click either **Active Jobs** or **Server Jobs**.
5. From the **View** menu, select **Customize this View**, then **Include**. The **Active Jobs - Include** dialog appears.



Active Jobs - Include

Include

Job:

Name: All

User: All

Number: All

Type

- [✓] Autostart
- [✓] Batch
- [✓] Communications
- [✓] Interactive
- [✓] Prestart
- [✓] Reader
- [✓] Subsystem
- [ ] System
- [✓] Writer

Subsystem: All

Current user: All

6. In the **Active Jobs - Include** dialog, select the options with which you want to search for your job.
7. Click **OK**. From this point, use **Find** to display a particular job.

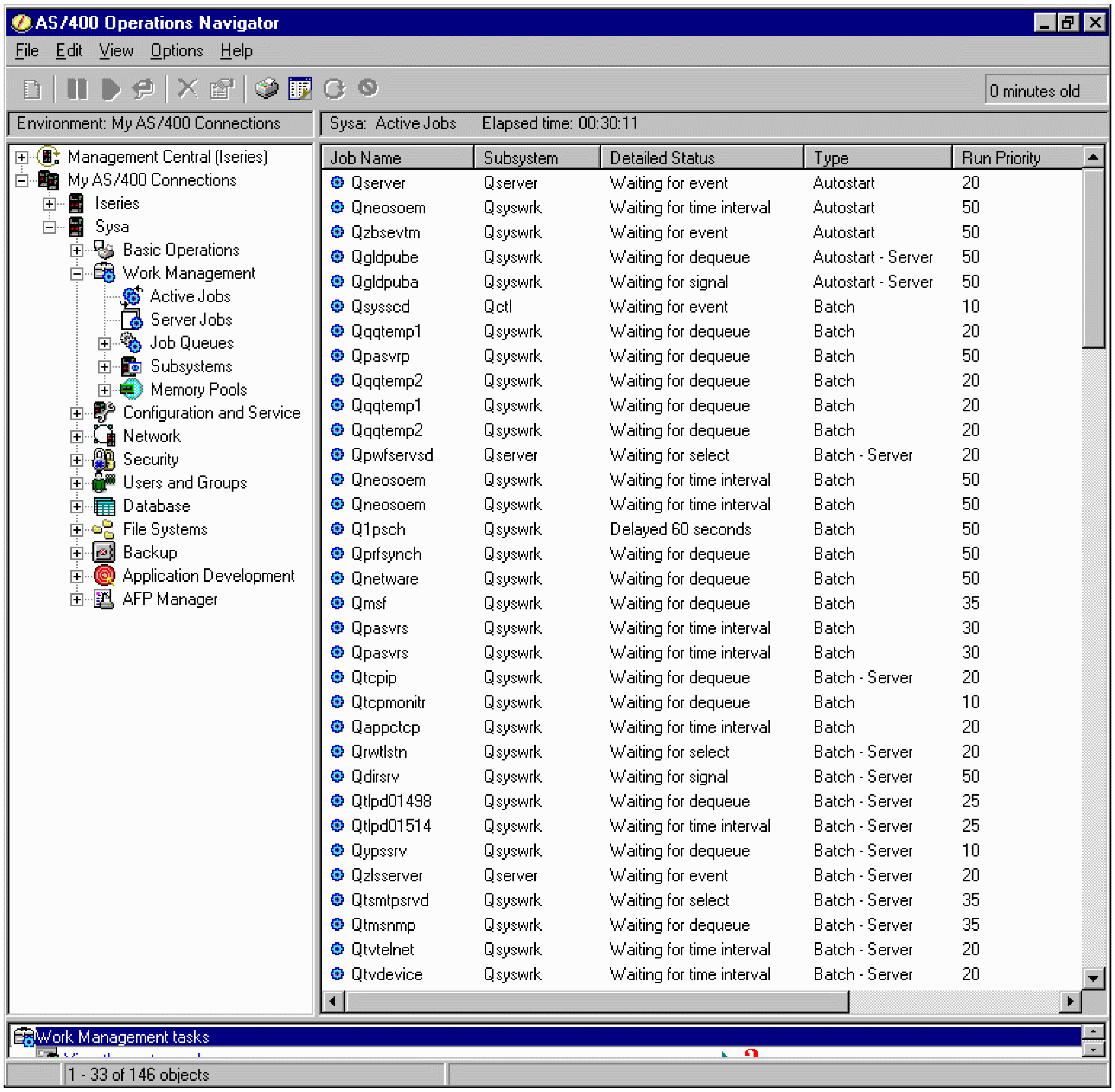
For more information on jobs, see Jobs.

Determine the status of a job

Monitoring your jobs will help you understand what your jobs are doing. The job status is an important piece of information that you can use to find out what a job is doing. In iSeries Navigator job status is easy to find.

To check the status of an active job or server job, do the following:

1. In iSeries Navigator, expand **My Connections**.
2. Expand the connection for your **iSeries server**.
3. Expand **Work Management**.
4. Click **Active Jobs** or **Server Jobs**.
   Note: You can see a job status from anyplace within the Work Management folder that you access jobs.
5. Look at the **Detailed Status** column to determine the status of a job (for example, Waiting for event, Waiting for time interval, or Waiting for dequeue).



Job Name	Subsystem	Detailed Status	Type	Run Priority
Qserver	Qserver	Waiting for event	Autostart	20
Qneosoem	Qsyswrk	Waiting for time interval	Autostart	50
Qzbsevtm	Qsyswrk	Waiting for event	Autostart	50
Qgldpube	Qsyswrk	Waiting for dequeue	Autostart - Server	50
Qgldpuba	Qsyswrk	Waiting for signal	Autostart - Server	50
Qsysscd	Qctl	Waiting for event	Batch	10
Qqqtemp1	Qsyswrk	Waiting for dequeue	Batch	20
Qpasvrp	Qsyswrk	Waiting for dequeue	Batch	50
Qqqtemp2	Qsyswrk	Waiting for dequeue	Batch	20
Qqqtemp1	Qsyswrk	Waiting for dequeue	Batch	20
Qqqtemp2	Qsyswrk	Waiting for dequeue	Batch	20
Qpwfservsd	Qserver	Waiting for select	Batch - Server	20
Qneosoem	Qsyswrk	Waiting for time interval	Batch	50
Qneosoem	Qsyswrk	Waiting for time interval	Batch	50
Q1psch	Qsyswrk	Delayed 60 seconds	Batch	50
Qprfsynch	Qsyswrk	Waiting for dequeue	Batch	50
Qnetware	Qsyswrk	Waiting for dequeue	Batch	50
Qmsf	Qsyswrk	Waiting for dequeue	Batch	35
Qpasvrs	Qsyswrk	Waiting for time interval	Batch	30
Qpasvrs	Qsyswrk	Waiting for time interval	Batch	30
Qtcpip	Qsyswrk	Waiting for dequeue	Batch - Server	20
Qtcpmonitr	Qsyswrk	Waiting for dequeue	Batch	10
Qappctcp	Qsyswrk	Waiting for time interval	Batch	20
Qrwtlstn	Qsyswrk	Waiting for select	Batch - Server	20
Qdirsrv	Qsyswrk	Waiting for signal	Batch - Server	50
Qtlpd01498	Qsyswrk	Waiting for dequeue	Batch - Server	25
Qtlpd01514	Qsyswrk	Waiting for time interval	Batch - Server	25
Qypssrv	Qsyswrk	Waiting for dequeue	Batch - Server	10
Qzlsserver	Qserver	Waiting for event	Batch - Server	20
Qtsmtpsrvd	Qsyswrk	Waiting for select	Batch - Server	35
Qtmsnmp	Qsyswrk	Waiting for dequeue	Batch - Server	35
Qtvtelnet	Qsyswrk	Waiting for time interval	Batch - Server	20
Qtvdevice	Qsyswrk	Waiting for time interval	Batch - Server	20

For more detailed information, see Job status.

View performance statistics for a job

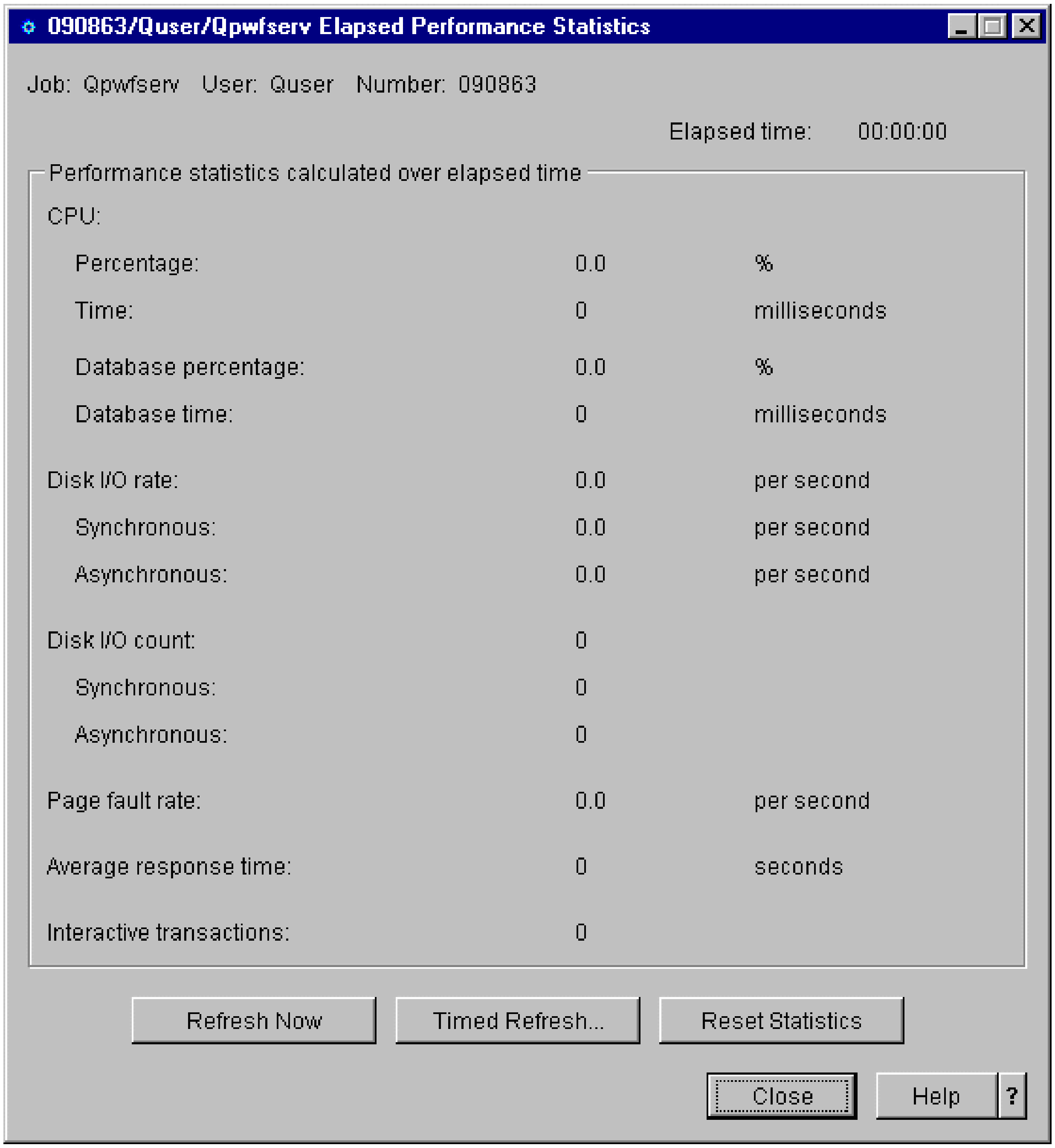
A job's performance is important to anyone that uses an iSeries server because one job running poorly can affect other jobs on the system. To view potentially problematic jobs gives you the ability to prevent performance problems before they occur.

The Elapsed Performance Statistics window allows you to monitor a job's CPU use, disk I/O (hard drive input/output), page fault rates, average response times, and the number of interactive transactions. You can select an option in this window to refresh these statistics manually or on a schedule.

To display the elapsed performance statistics, do the following:

1. In iSeries Navigator, expand **My Connections**.
2. Expand the connection for your **iSeries server**.
3. Expand **Work Management**.
4. Click **Active Jobs**.
   Note: You can view the performance of a job from any location within work management where you can see jobs. The **Elapsed Performance Statistics** dialog can be displayed from the Performance tab of a **Job** property sheet.
5. Right-click the job for which you want to display the performance statistics, and select **Details...**.

6. From the **Details...** list, select **Elapsed Performance Statistics**.



090863/Quser/Qpwfserv Elapsed Performance Statistics

Job: Qpwfserv User: Quser Number: 090863

Elapsed time: 00:00:00

Performance statistics calculated over elapsed time

CPU:		
Percentage:	0.0	%
Time:	0	milliseconds
Database percentage:	0.0	%
Database time:	0	milliseconds
Disk I/O rate:	0.0	per second
Synchronous:	0.0	per second
Asynchronous:	0.0	per second
Disk I/O count:	0	
Synchronous:	0	
Asynchronous:	0	
Page fault rate:	0.0	per second
Average response time:	0	seconds
Interactive transactions:	0	

You can refresh, reset, and schedule the performance statistics to automatically refresh.

Note: You can look at the elapsed performance statistics for more than one job at a time by opening multiple windows. This allows you to view multiple problematic jobs at one time. Each window holds the information for only one job.

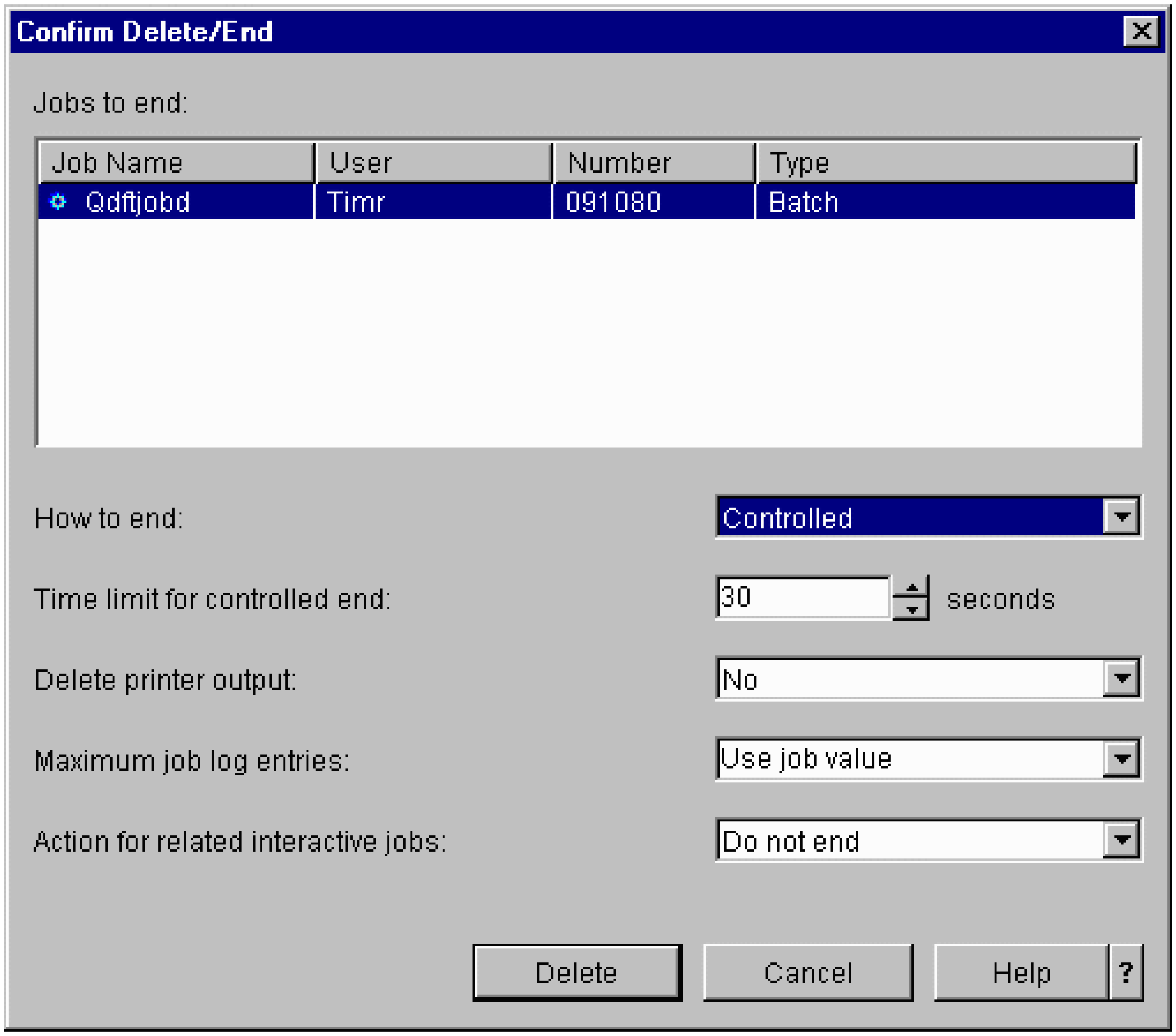
The elapsed performance statistics is one way to view the performance of a job as it moves through the system. Another way to view jobs on the system is through the Management Central folder. You can monitor jobs in Management Central as well as monitor system performance and messages. For additional information on job monitors, see Monitoring jobs and servers with Management Central.

End a job

Sometimes you need to end jobs because they take too long to run or they use too much memory, which can affect the performance of other jobs on the system.

To end a job, do the following:

1. In iSeries Navigator, expand **My Connections**.
2. Expand the connection for your **iSeries server**.
3. Expand **Work Management**.
4. Click **Active Jobs**.
   Note: You can **Delete/End** a job from any location within work management where you can see jobs.
5. Right-click the job that you want to end (for example, Qdftjobd) and click **Delete/End**.



Jobs to end:

Job Name	User	Number	Type
Qdftjobd	Timr	091080	Batch

How to end: Controlled

Time limit for controlled end: 30 seconds

Delete printer output: No

Maximum job log entries: Use job value

Action for related interactive jobs: Do not end

6. In the **How to end** field, select **Controlled** or **Immediate**.
7. In the **Time limit for controlled end** field, enter the number of seconds before the job switches from a controlled end to an immediate end. (This parameter only applies to a controlled Delete/End.)
8. In the **Delete printer output** field, select **Yes** or **No**.

9. In the **Maximum job log entries** field, select **Use job value** or **No maximum**.
10. In the **Action for related interactive jobs** field, choose **Do not end, End for group jobs,** or **End all**.
11. Click **Delete** to delete the job.

For more information on the actions you can perform on jobs, see Job actions.

Job actions

Managing jobs and threads is made more efficient with the actions available in Work Management. Once you find the job you want to manage, the following actions are available by right-clicking the job:

Reset statistics
Allows you to reset the list information you are viewing, and it sets the elapsed time to 00:00:00.

Printer output
Displays printer output, if available, in a separate window.

Job log
Displays the job log for the selected job, in a separate window.

Details
Contains detailed information about the following actions for active jobs:

- Call stack
- Library list
- » Locked objects «
- Open files
- » Threads «
- » Transactions «
- Elapsed performance statistics
- » Last SQL statement «

Reply
Allows you to reply to the message, if you have a job that is waiting for a message.

Hold
Allows you to hold the job. Holding a job holds all threads in the job. This is available for released jobs that are not system jobs. When you hold a job, the job is not available for processing. An active job can be held to temporarily stop its processing.

Release
Releases the job that was held. Releasing the job releases all threads in the job that were held with the **Hold** job action. The job is made available for processing.

Move
Allows you to move the selected job to another job queue. You can only move jobs that are on a job queue.

Delete/End
Allows you to end the selected job. The two ways to end a job, either controlled or immediately.

Monitor
Allows you to create a job monitor for one or more jobs.

Job properties
The job properties for the selected job can be viewed and changed.

View threads running under a specific job

Every active job running on an iSeries system has at least one thread running under it. A thread is an independent unit of work running within a job that uses the same resources as the job. Because a job depends on the work done by a thread, it is important to know how to find the threads running within a specific job.

To view threads running under a specific job, do the following:

1. In iSeries Navigator, expand **My Connections**.
2. Expand the connection for your **iSeries server**.
3. Expand **Work Management**.
4. Click **Active Jobs**.
5. Right-click the job with which you want to work, and select **Details** > Threads.

For more detailed information, see Threads or see the iSeries Navigator help.

View thread properties

Threads allow jobs to do more than one thing at a time. If a thread stops processing, it can stop the job from running. The Thread Properties pages allow you to view various thread and thread performance properties that can aid in understanding why a thread is not running.

To view the properties of a thread, do the following:

1. In iSeries Navigator, expand **My Connections**.
2. Expand the connection for your **iSeries server**.
3. Expand **Work Management**.
4. Click **Active Jobs** or **Server Jobs**.
5. Right click the job with which you want to work, and select **Details** > **Threads**.
6. Right-click the thread with which you want to work, and select **Properties**.

For more detailed information, see Threads or see the iSeries Navigator help.

Delete or end a thread

An initial thread, which is created when the job starts, can never be deleted or ended. However, sometimes it is necessary to end a secondary thread so that a job can continue to run. Be aware of the thread you intend to end because the job it runs within may not be able to complete without that thread's work.

Important:	Ending threads should not be a part of your daily work management routine. Ending a thread is more serious than ending a job because the work in other threads may or may not stop. When you end a job, all the work stops. However, when you end a thread, only a portion of the work stops. Other threads may or may not continue to run. If they continue running without the thread that you end, they may produce undesirable results.

To delete or end a secondary thread, you must have service (*SERVICE) special authority or Thread Control authority.

Thread Control authority allows a user to end, hold, and release threads of another job. It allows one to retrieve information about threads of another job. Thread Control can be granted and revoked for individual users by using iSeries Navigator's Application Administration support, or by using the Change Function Usage Information (QSYCHFUI) API, with a function ID of QIBM_SERVICE_THREAD. For more detailed information, see Application Administration.

To delete or end a thread, do the following:

1. In iSeries Navigator, expand **My Connections**.
2. Expand the connection for your **iSeries server**.
3. Expand **Work Management**.
4. Click **Active Jobs** or **Server Jobs**.
5. Right-click the job with which you want to work, and select **Details**, and then **Threads**.
6. Right-click the thread with which you want to end, and select **Delete/End**.

For more detailed information, see Threads or see the iSeries Navigator help.

Manage job queues

In the life cycle of a batch job, job queues are the entry point into the subsystem. Job queues manage the number of jobs allowed into the subsystem at any given time and the order they are allowed into the subsystem.

These subtopics provide instructions for the following tasks:

- View jobs on the job queue
- Change the priority of a job within a job queue
- Move jobs to different job queues

For more information, see Job queues.

View jobs on the job queue

Job queues filter some of the work that is processed in work management (for example, some batch jobs). Being able to view jobs in the job queue allows you to see what jobs are waiting to be sent to a subsystem.

To view jobs on the job queue, do the following:

1. In iSeries Navigator, expand **My Connections**.
2. Expand the connection for your **iSeries server**.
3. Expand **Work Management**.
4. Expand **Job Queues**.
5. Expand **Active Job Queues**. You can also choose to expand **All Job Queues**.
6. Select the job queue with which you want to display the jobs (for example, Jobqueue1). The jobs within the job queue appear.

For more information, see Job queues.

Change the priority of a job within a job queue

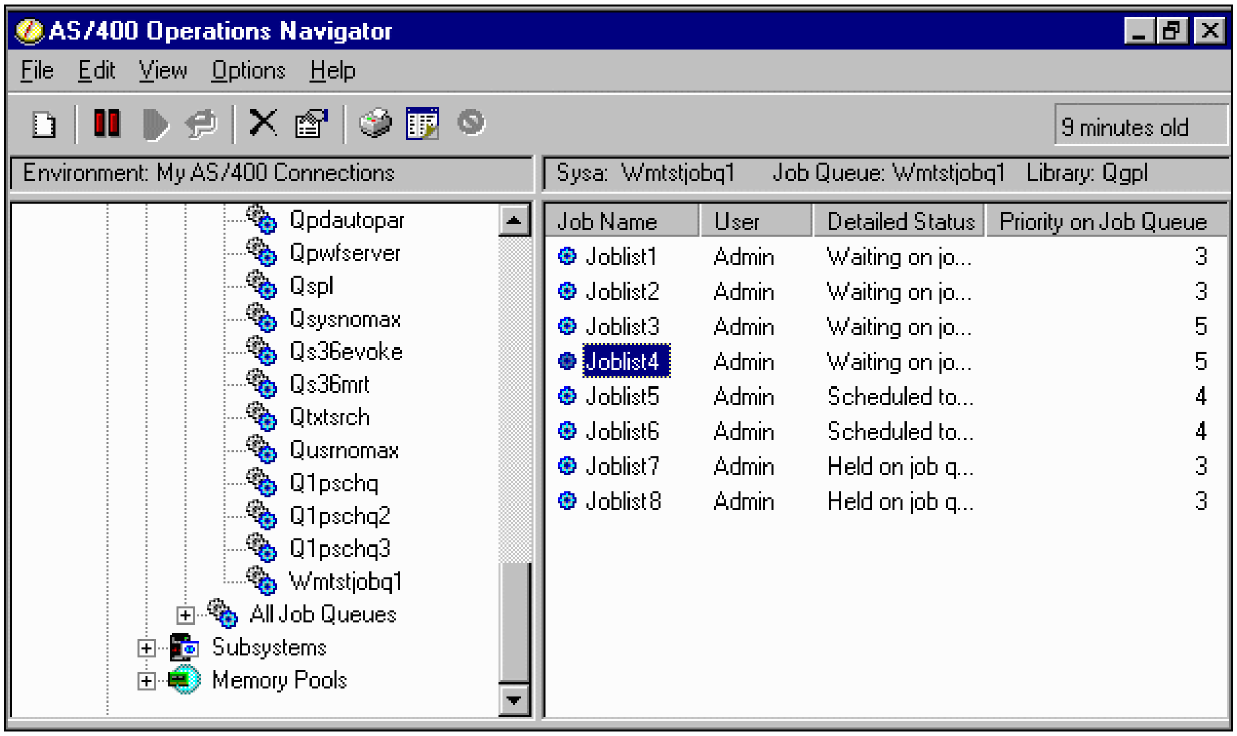
Sometimes the importance of a job changes as it goes through its life cycle. It can increase or decrease in priority in relation to other jobs. Because these changes occur, you need to know how to change the priority of a job within the job queue. The priority of a job on a job queue helps determine when the job goes to the subsystem to run. A range from zero to nine (zero being the most important) determines the priority of a job on a job queue.

Within iSeries Navigator, you can either drag and drop jobs or use the property page to increase or decrease the priority of a job.

To change the job queue priority of a job on a job queue using drag and drop, do the following:

1. In iSeries Navigator, expand **My Connections**.
2. Expand the connection for your **iSeries server**.
3. Expand **Work Management**
4. Expand **Job Queues**.
5. Expand either **Active Job Queues** or **All Job Queues**. A list of job queues appears in the right pane.
6. Select the job queue you want to work in (for example, Qbatch). A list of the jobs on the job queue appears.

7. Click the job for which you want to move, and drag it to the new priority position (for example, you want to move joblist4 with a priority of 5 after joblist1 which has a priority of 3).



Sysa: Wmtstjobq1 Job Queue: Wmtstjobq1 Library: Qgpl

Job Name	User	Detailed Status	Priority on Job Queue
Joblist1	Admin	Waiting on jo...	3
Joblist2	Admin	Waiting on jo...	3
Joblist3	Admin	Waiting on jo...	5
Joblist4	Admin	Waiting on jo...	5
Joblist5	Admin	Scheduled to...	4
Joblist6	Admin	Scheduled to...	4
Joblist7	Admin	Held on job q...	3
Joblist8	Admin	Held on job q...	3

Use the property page to change the job queue priority of a job on a job queue:

1. In iSeries Navigator, expand **My Connections**.
2. Expand the connection for your **iSeries server**.
3. Expand **Work Management**
4. Expand **Job Queues**.
5. Expand either **Active Job Queues** or **All Job Queues**. A list of job queues appears in the right pane.
6. Select the job queue you want to work in (for example, Qbatch). A list of the jobs on the job queue appears.
7. Right-click the job for which you want to change the priority and select **Properties**. The **Properties** dialog appears.
8. Click the **Job Queue** tab.
9. From the **Priority on job queue** list, select a higher (or lower) priority number. The job queue priority ranges from 0-9, with 0 being the highest priority.
10. Click **OK**. The job queue priority has been changed for your job. For example, changing a priority 4 job to a priority 3 moves the job to the bottom of the list of jobs that have a priority 3.
11. Press **F5** to refresh the Job Queue window.

For more information, see Job queues.

Move jobs to different job queues

Sometimes you need to move jobs from one job queue to another job queue, whether it is because a job queue is too congested and the jobs are not moving quickly to the subsystem or because you create a special job queue for important jobs. iSeries Navigator makes moving jobs between job queues quick and easy.

A job can be moved from one job queue to another job queue in one of two ways, use either drag and drop or the **Move Job** dialog.

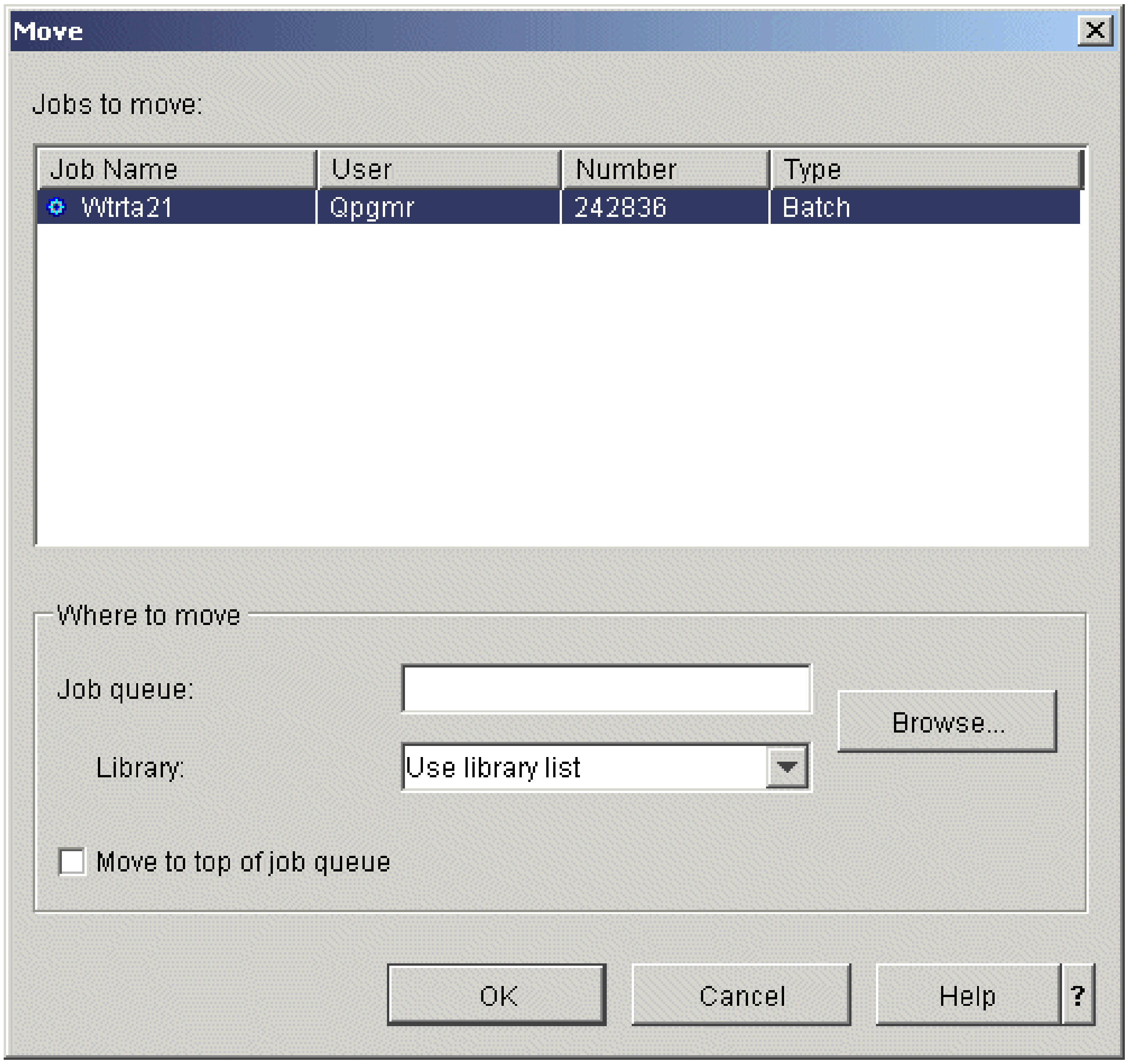
To drag and drop a job from one job queue to another job queue, do the following:

1. In iSeries Navigator, expand **My Connections**.
2. Expand the connection for your **iSeries server**.
3. Expand **Work Management**.
4. Select **Job Queues**.
5. Click **Active Job Queues** or **All Job Queues**.
6. Double-click the job queue with which you want to work.
7. Select the job you want to move.
   Note: You can select multiple jobs to move to another job queue by pressing Ctrl+Shift and selecting each job you want to move.
8. Drag the job to the desired job queue. When the job or jobs are dropped on a new job queue, the job or jobs are put into the same relative position they were in on their previous job queue. For example, a priority 3 job that is moved to a new job queue is placed at the end of the priority 3 jobs in the new job queue.
   Note: If you drag using the right mouse button, a menu appears with the commands **Move**, **Move to Top**, and **Cancel**. Click the command you want.

To use the **Move...** dialog to move a job from one job queue to another job queue, do the following:

1. In iSeries Navigator, expand **My Connections**.
2. Expand the connection for your **iSeries server**.
3. Expand **Work Management**.
4. Select **Job Queues**.
5. Click **Active Job Queues** or **All Job Queues**.
6. Click the job queue with which you want to work in.

7. Right-click the job you want to move to another job queue (for example, Qdftjobd) and select **Move...**. **Note:** You can select multiple jobs to move from one job queue to another job queue.


Move

Jobs to move:

Job Name	User	Number	Type
Wtrta21	Qpgmr	242836	Batch

Where to move

Job queue:

Library: Use library list

☐ Move to top of job queue

8. In the **Jobs to move** field, verify that your job is highlighted. If you want to remove selected jobs, you can press Ctrl and click the jobs you want to remove.
9. In the **Where to move Job Queue** field, type or browse to the job queue where you want to move your job (for example, Qusrnomax).
10. In the **Library field**, type the name of the job queue library or select from the available list.
11. Click **OK**.

    When the job or jobs are moved to a new job queue, the job or jobs are put into the same relative position they were in on their previous job queue. For example, a priority 3 job that is moved to a new job queue is placed at the end of the priority 3 jobs in the new job queue. If a job that is held is moved, the job remains held and is placed in the same relative position in the new job queue.

    By checking the **Move to Top** box, the job is moved to the top of the target queue, without regard to its current status and priority. (However, if the job at the top of the target queue has a priority greater than the user is allowed, an error message is displayed and the job is not moved.) Jobs that are waiting to run can be moved to the top of another queue. For example, if the selected job has a job queue priority of 5 and the first job on the target queue has a priority of 3, the priority of the selected job is changed to 3 and is placed ahead of the other jobs on the target queue.

Jobs that are held are released and then moved to the top of the target queue. Jobs that are scheduled to run cannot be moved to the top of another queue. An error message is displayed stating that the selected job is not available to be moved.

For more information, see job queues.

Manage subsystems

The subsystem is the work place for jobs on the iSeries server. All user work is done by jobs running in the subsystem and it is important to monitor this area for slow work performance. In iSeries Navigator, you can view jobs and job queues associated with the subsystems. Also, you have the same functionality with jobs and job queues from any other area that displays jobs and job queues.

To learn more about subsystems, see these topics:

- Monitor a subsystem
- View jobs in a subsystem
- Start a subsystem
- Stop a subsystem

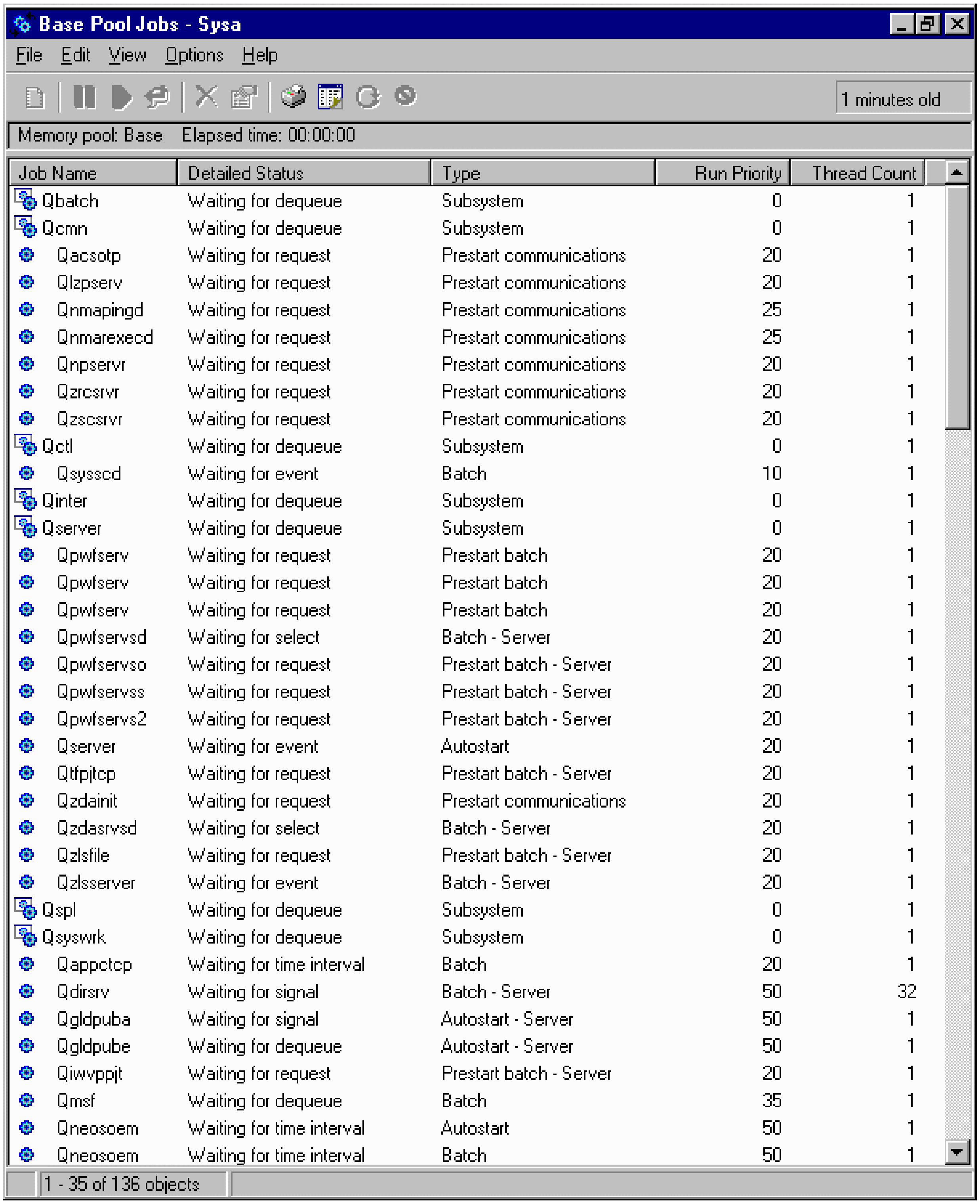
Monitor the number of jobs in a memory pool

Since memory pools give subsystems memory to run jobs, it is important to check on the number of jobs running in a memory pool. Too many jobs in one memory pool can negatively impact system performance.

To monitor the number of jobs in a memory pool, do the following:

1. In iSeries Navigator, expand **My Connections**.
2. Expand the connection for your **iSeries server**.
3. Expand **Work Management.**
4. Expand **Memory Pools**, and then click **Active Pools** or **Shared Pools**.
5. Right-click the memory pool you want to use (for example, Base) and select **Jobs**. A dialog appears showing a list of jobs within the memory pool.

You can also view the number of threads in a memory pool by viewing the Thread Count column. The thread count provides additional information about the amount of activity in a memory pool.



Memory pool: Base   Elapsed time: 00:00:00

Job Name	Detailed Status	Type	Run Priority	Thread Count
Qbatch	Waiting for dequeue	Subsystem	0	1
Qcmn	Waiting for dequeue	Subsystem	0	1
Qacsotp	Waiting for request	Prestart communications	20	1
Qlzpserv	Waiting for request	Prestart communications	20	1
Qnmapingd	Waiting for request	Prestart communications	25	1
Qnmarexecd	Waiting for request	Prestart communications	25	1
Qnpservr	Waiting for request	Prestart communications	20	1
Qzrcsrvr	Waiting for request	Prestart communications	20	1
Qzscsrvr	Waiting for request	Prestart communications	20	1
Qctl	Waiting for dequeue	Subsystem	0	1
Qsysscd	Waiting for event	Batch	10	1
Qinter	Waiting for dequeue	Subsystem	0	1
Qserver	Waiting for dequeue	Subsystem	0	1
Qpwfserv	Waiting for request	Prestart batch	20	1
Qpwfserv	Waiting for request	Prestart batch	20	1
Qpwfserv	Waiting for request	Prestart batch	20	1
Qpwfservsd	Waiting for select	Batch - Server	20	1
Qpwfservso	Waiting for request	Prestart batch - Server	20	1
Qpwfservss	Waiting for request	Prestart batch - Server	20	1
Qpwfservs2	Waiting for request	Prestart batch - Server	20	1
Qserver	Waiting for event	Autostart	20	1
Qtfpjtcp	Waiting for request	Prestart batch - Server	20	1
Qzdainit	Waiting for request	Prestart communications	20	1
Qzdasrvsd	Waiting for select	Batch - Server	20	1
Qzlsfile	Waiting for request	Prestart batch - Server	20	1
Qzlsserver	Waiting for event	Batch - Server	20	1
Qspl	Waiting for dequeue	Subsystem	0	1
Qsyswrk	Waiting for dequeue	Subsystem	0	1
Qappctcp	Waiting for time interval	Batch	20	1
Qdirsrv	Waiting for signal	Batch - Server	50	32
Qgldpuba	Waiting for signal	Autostart - Server	50	1
Qgldpube	Waiting for dequeue	Autostart - Server	50	1
Qiwvppjt	Waiting for request	Prestart batch - Server	20	1
Qmsf	Waiting for dequeue	Batch	35	1
Qneosoem	Waiting for time interval	Autostart	50	1
Qneosoem	Waiting for time interval	Batch	50	1

1 - 35 of 136 objects

From this point, you can perform the same functions on jobs as if you were in the Active jobs or Server jobs area.

For more information, see Memory pools.

View jobs in the subsystem

Subsystems coordinate work flow and the resources that a job uses to run. iSeries Navigator allows you to see what jobs are currently active (but not necessarily running) in the subsystem.

To view jobs in the subsystem, follow these steps:
1. In iSeries Navigator, expand **My Connections**.
2. Expand the connection for your **iSeries server**.
3. Expand **Work Management**.
4. Expand **Subsystems**.
5. Expand **Active Subsystems**, and then select the subsystem for which you want to display its jobs.

For more information, see Subsystems.

Start a subsystem

When a subsystem is started, the system allocates the available resources that are defined to it in the subsystem description such as memory pools, workstations, and job queues. These resources prepare the subsystem for use.

For details on the chain of events that are triggered when a subsystem starts, see what happens when the subsystem starts.

To start a subsystem, follow these steps:
1. In iSeries Navigator, expand **My Connections**.
2. Expand the connection for your **iSeries server**.
3. Expand **Work Management**.
4. Right-click **Subsystems**, and then select **Start Subsystem**.
5. Specify the name and the library for the subsystem to be started, or click **Browse...** to select from a list of subsystems.
6. Click **OK**.

Stop a subsystem

You can use iSeries Navigator to stop one or more active subsystems and specify what happens to active work being processed. No new jobs or routing steps are started in the subsystem after the subsystem is stopped.

When a subsystem is stopped, you can specify what happens to active work that is being processed by the system. For example, you can specify for all jobs in the subsystem to be ended immediately (**Immediate**), or you can specify that jobs are allowed to finish processing before the subsystem ends (**Controlled**).

> **Important**: It is recommended that subsystems be stopped using the **Controlled** option whenever possible. This allows active jobs to end themselves. Use this option to ensure that jobs finish before the subsystems end. This allows the programs that are running to perform cleanup (end-of-job processing). Specifying the **Immediate** value can cause undesirable results, for example, from data that has been partially updated.

There are additional options available when stopping subsystems. These options are described in detail in the help associated with the **Stop Subsystem** dialog in iSeries Navigator.

To stop a subsystem, follow these steps:
1. In iSeries Navigator, expand **My Connections**.
2. Expand the connection for your **iSeries server**.
3. Expand **Work Management**.

4. Expand **Active Subsystems**.
5. Right-click the subsystem or subsystems you would like to stop, and then select **Stop...**.
6. Specify the options to be used when the subsystem is stopped.
7. Click **Stop**.

Manage memory pools

Memory pools allocate memory that subsystems use to run jobs. If too much memory is given to one subsystem and not enough to another subsystem, jobs in the subsystem begin to run poorly. The iSeries server provides a default tuner that will meet the needs of many users. However, if your requirements exceed the capabilities of the system tuner, you will want to know how to manage your memory pools. You can access the performance tuning values in iSeries Navigator by going through the Properties for a shared memory pool to the **Tuning** page. For more information, see Performance. If you want more information on how to tune performance on your system, see Tune performance.

To manage memory pools, see these topics:

- Monitor the number of jobs in a memory pool
- Monitor the number of subsystems using a memory pool
- Check memory pool use
- Change the size of a memory pool

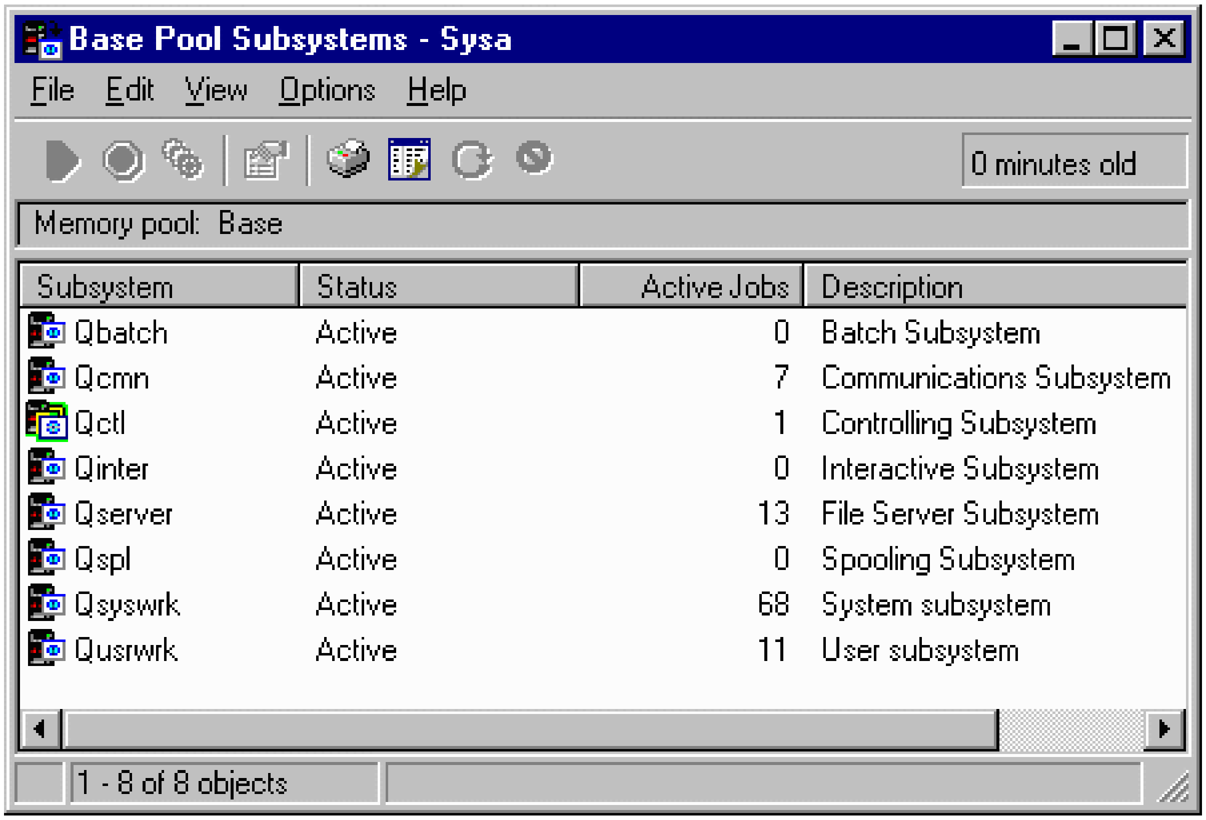
Monitor the number of subsystems using a memory pool

Subsystems are allocated a certain percentage of memory to run jobs. It is important, as far as performance, to know how many different subsystems are pulling from the same memory pool. Once you know how many subsystems are submitting jobs to a pool and how many jobs are running in a pool, you may want to adjust the size and activity level of the pool to reduce resource contention.

To monitor the number of subsystems using a memory pool, do the following:

1. In iSeries Navigator, expand **My Connections**.
2. Expand the connection for your **iSeries server**.
3. Expand **Work Management**.
4. Expand **Memory Pools**.
5. Click **Active Pools** or **Shared Pools**.

6. Right-click the memory pool you want to work with and select **Subsystems** (for example, Base).



Memory pool: Base

Subsystem	Status	Active Jobs	Description
Qbatch	Active	0	Batch Subsystem
Qcmn	Active	7	Communications Subsystem
Qctl	Active	1	Controlling Subsystem
Qinter	Active	0	Interactive Subsystem
Qserver	Active	13	File Server Subsystem
Qspl	Active	0	Spooling Subsystem
Qsyswrk	Active	68	System subsystem
Qusrwrk	Active	11	User subsystem

1 - 8 of 8 objects

From this window, you can determine the number of subsystems that are using an individual memory to run their jobs.

For more information, see Memory pool activity level.

Check memory pool use

Periodically checking the amount of memory your memory pools use is important. By monitoring these levels, you can tune your pools to run at maximum efficiency, which in turn, keeps the work cycle running smoothly. In iSeries Navigator, you can easily monitor the amount of memory your pools are using.

To check the memory use, do the following:

1. In iSeries Navigator, expand **My Connections**.
2. Expand the connection for your **iSeries server**.
3. Expand **Work Management**.
4. Expand **Memory Pools**, and then click **Active Pools** or **Shared Pools**.
5. Right-click the memory pool you want to work with (for example, Interactive) and select **Properties**.
6. Click the **Configuration** tab. The **Current** field, under Size, shows the amount of memory the pool currently has.

   Note: You can also view the current size of a memory pool when you click **Active Pools** or **Shared Pools**. Current Size (in megabytes) is a default column you see when a list of memory pools appears in the right pane of iSeries Navigator.

For more information, see Memory pools.

Change the size of a memory pool

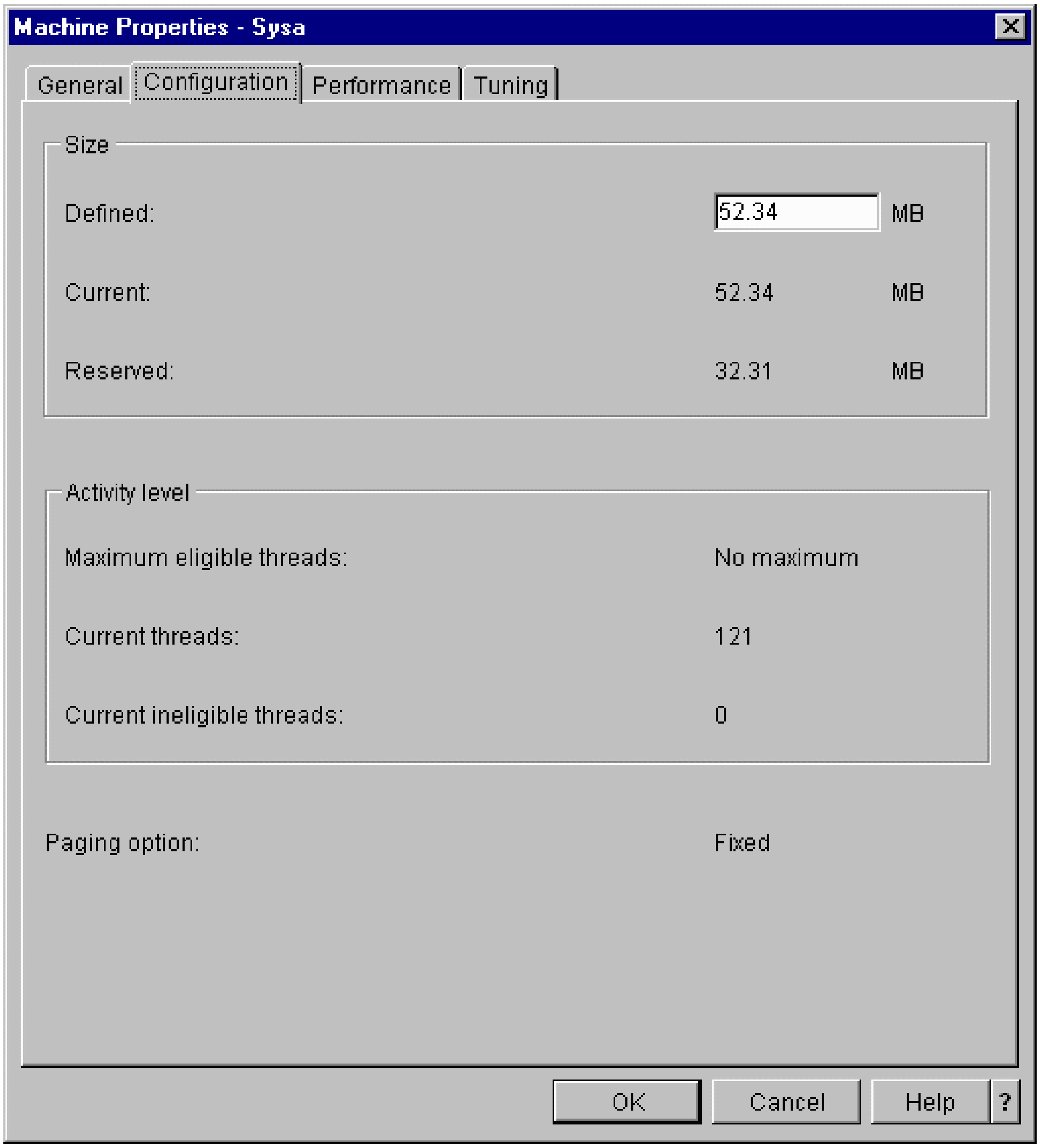
The size of a memory pool directly affects the amount of work a subsystem can process. The more memory it has, the more work a subsystem can potentially complete. In iSeries Navigator, you can change the amount of defined (or available) memory a pool has. However, it is important that you monitor your system carefully before you start changing the parameters of your memory pools. You will also want to periodically recheck these levels, as some readjustment may need to be done.

Note: Make sure you turn off the system tuner before you start manually changing memory pool sizes. The system tuner automatically adjusts the sizes of your shared memory pools to the amount of work the system is doing. If the system tuner is not turned off, the changes you make manually may be changed automatically by the tuner.

To change the size of a memory pool, do the following:

1. In iSeries Navigator, expand **My Connections**.
2. Expand the connection for your **iSeries server**.
3. Expand **Work Management**.
4. Expand **Memory Pools**, and then click **Active Pools** or **Shared Pools**.
5. Right-click the memory pool you want to work in (for example, Interactive) and select **Properties**. The Memory Pool Properties window appears.

6. Click the **Configuration** tab.



Size

Defined: 52.34 MB

Current: 52.34 MB

Reserved: 32.31 MB

Activity level

Maximum eligible threads: No maximum

Current threads: 121

Current ineligible threads: 0

Paging option: Fixed

From the Configuration tab of the Properties window, you can change the defined amount of memory. Defined memory is the maximum amount of memory that that pool can use. The number you put here should reflect the amount of memory you think that pool will need to support the subsystems it services.

Special considerations for Base pool: The Base pool is the only memory pool that does not have a defined amount of memory. It has a minimum amount of memory that it needs to run. The Base pool contains everything that is not allocated elsewhere. For example, you may have 1000 MB of memory on your system of which 250 MB is allocated to the Machine pool and 250 MB is allocated to the Interactive pool. 500 MB not allocated to anything. This nonallocated memory is stored in the Base pool until it is needed. Use caution when moving memory. Moving memory from one pool to another can fix one subsystem, but can cause problems for other subsystems, which in turn, can worsen system performance.

For more information, see Memory pools.

Manage job logs

Most jobs on your iSeries have a job log associated with it. Job logs tell the user many different things such as when the job starts, when the job ends, what commands are running, failure notices and error messages. This information gives the user a good idea of how the job cycle is running.

Find out how to access the job log of an active job and access the job log printer output.

- Accessing job logs for active jobs, including server jobs
- Accessing job log printer output

For more information, see Job logs in Chapter 5 of the Work Management

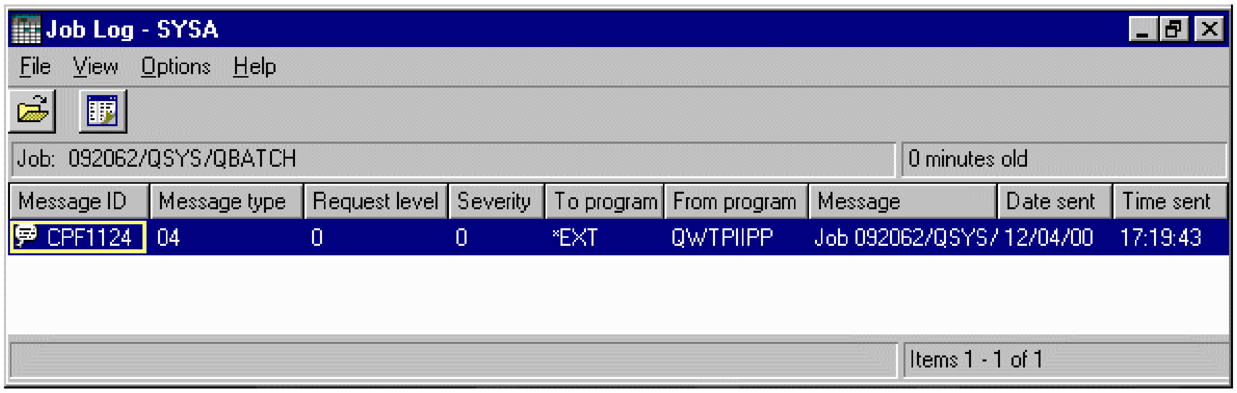
manual.

Access job logs for active jobs, including server jobs

Because job logs record information about a job while it is running, it is important to know how to access them.

To access the job log for an active job or server job, do the following:

1. In **iSeries Navigator**, expand **My Connections**.
2. Expand the connection for your **iSeries server**.
3. Expand **Work Management**.
4. Select **Active Jobs** or **Server Jobs**.
   Note: You can see a job log from any place within work management that you access jobs (for example, through the Subsystem area or the Memory Pool area).
5. Right-click a job (for example, Qbatch) and select **Job Log**. Use the image below to see the types of information you can find in a job log. For more information, refer to the help in the **Job Log** dialog.



Job: 092062/QSYS/QBATCH | 0 minutes old

Message ID	Message type	Request level	Severity	To program	From program	Message	Date sent	Time sent
CPF1124	04	0	0	*EXT	QWTPIIPP	Job 092062/QSYS/	12/04/00	17:19:43

Items 1 - 1 of 1

To view more details of a message, double-click a specific message. A **Detailed Message Information** dialog appears. This dialog shows the details of the message as well as the message help. The detailed message help gives you information to solve a problem.

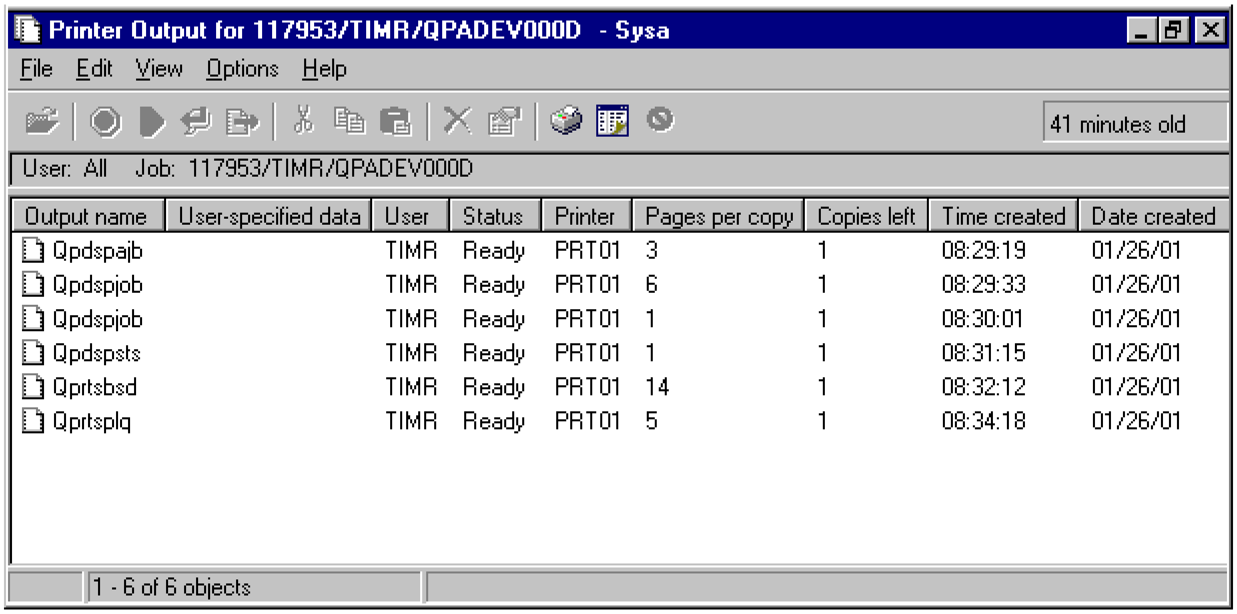
For more information, see Job logs or refer to the help.

Access printer output

Because you have the choice to detach printer output from a job once it finishes running (separating the printer output from the job completely), you can access your printer output in iSeries Navigator through Basic Operations or through Work Management.

To access a job's printer output through Basic Operations, do the following:

1. In iSeries Navigator, expand **My Connections**.
2. Expand the connection for your **iSeries server**.
3. Expand **Basic Operations**.
4. Select **Job**. All jobs for the current user appear. See Find a job on the iSeries server for the different ways to search for jobs.
5. Right-click the job for which you want to display printer output and click **Printer Output**. The **Printer Output** dialog appears.


Printer Output for 117953/TIMR/QPADEV000D - Sysa
File Edit View Options Help
41 minutes old

User: All Job: 117953/TIMR/QPADEV000D

Output name	User-specified data	User	Status	Printer	Pages per copy	Copies left	Time created	Date created
Qpdspajb		TIMR	Ready	PRT01	3	1	08:29:19	01/26/01
Qpdspjob		TIMR	Ready	PRT01	6	1	08:29:33	01/26/01
Qpdspjob		TIMR	Ready	PRT01	1	1	08:30:01	01/26/01
Qpdspsts		TIMR	Ready	PRT01	1	1	08:31:15	01/26/01
Qprtsbsd		TIMR	Ready	PRT01	14	1	08:32:12	01/26/01
Qprtsplq		TIMR	Ready	PRT01	5	1	08:34:18	01/26/01

1 - 6 of 6 objects

To access printer output through the **Output Queues** folder, do the following:

1. In iSeries Navigator, expand **My Connections**.
2. Expand the connection for your **iSeries server**.
3. Expand **Work Management**.
4. Expand **Output Queues**.
5. Select the output queue with which you want to display printer output (for example, Qprint2). The printer output within the output queue appears.

Manage output queues

Printer output resides on the output queue. The output queue determines the order in which printer output will be processed by the print device. By managing your output queues, you can ensure smooth processing of your printer output.

With the proper authority, you can complete the following tasks from the **Output Queues** folder:

- View output queues on the system
- View the properties of an output queue
- Hold an output queue
- Release an output queue

- Clear an output queue
- View output waiting on an output queue
- Move output between and within an output queue
- Change the properties of an output queue

Use these subtopics to view output queues on your system, clear output queues, and move printer output between and within output queues.
- View output queues on the system
- Move output between and within output queues
- Clear output queues

For more information on the different tasks you can complete with output queues, see the iSeries Navigator online help. For more information, see Output Queues.

View output queues on the system

Output queues determine the order in which printer output is sent to the printer device.

To view output queues on the system, do the following:
1. In iSeries Navigator, expand **My Connections**.
2. Expand the connection for your **iSeries server**.
3. Expand **Work Management**.
4. Expand **Output Queues**.

In iSeries Navigator, you can customize the list of output queues you are viewing by using the Include... dialog. The Include... dialog allows you to put limitations on what is displayed in iSeries Navigator. For example, you can run Include... to display only certain output queues. To use the include function, use the **View** menu, and then **Customize this View**.

For more information, see Output queues .

Move output between and within output queues

Sometimes you need to move your output from one queue to another queue or you need to move it to a higher priority level so that it is sent to the printer device more quickly. This can happen if too much output traffic is on an output queue.

You can move output from one output queue to another or you can move output within an output queue.

To move output between output queues, follow these steps:
1. In iSeries Navigator, expand **My Connections**.
2. Expand the connection for your **iSeries server**.
3. Expand **Work Management**.
4. Expand **Output Queues**.
5. Double-click the output queue that contains the output you would like to move.
6. Click the output you would like to move, and drag it to the output queue to which you would like to move it in the left pane of iSeries Navigator.

Note: The output is moved to the target queue and placed on the queue according to priority.

To move output within an output queue, follow these steps:

1. In iSeries Navigator, expand **My Connections**.
2. Expand the connection for your **iSeries server**.
3. Expand **Work Management**.
4. Expand **Output Queues**.
5. Double-click the output queue that contains the output you would like to move.
6. Click the output you would like to move, and drag it to the output in the queue that you would like to move it after.

Note: The output is moved directly after the target output.

For more information, see Output Queues.

Clear output queues

When a job creates printer output it is sent to an output queue to be printed. Most likely you will not print all the printer output created. iSeries Navigator gives you the ability to clean out your output queues using the **Clear** option. Clearing an output queue will delete all output from the queue.

To clear an output queue, follow these steps:

1. In iSeries Navigator, expand **My Connections**.
2. Expand the connection for your **iSeries server**.
3. Expand **Work Management**.
4. Expand **Output Queues**.
5. Right-click the output queue you would like to clear, and select **Clear**.

For more information, see Output queues.

The structure of your system

You can separate work management into five different functional areas: jobs, job queues, subsystems, memory pools, and output. Each of these areas has its own terms and concepts associated with it. By themselves they produce different types of data; however, when integrated with each other they become a powerful tool for managing work on your iSeries server.

To learn more about the different functional areas within work management, see these topics:

Jobs
Learn about the different types of jobs and their properties. Also, learn about the actions that you can perform on jobs.

Job queues
Learn about the role of the job queue in the work management life cycle.

Subsystems
Learn about the different types of subsystems and their properties.

Memory pools
Learn about the different types of memory pools and their properties.

Output queues
Learn what happens to work when it finishes running.

Note: iSeries Navigator calls application programming interfaces (APIs) that retrieve information from the iSeries system. APIs are iSeries Navigator's input and output devices for the iSeries server. For more information on APIs, see Application programming interfaces (APIs) or the System API Programming

.

Jobs

All work done on a system is performed through jobs. Each active job contains at least one thread (the initial thread) and may contain additional secondary threads. Threads are independent units of work. Job properties are shared among the threads of the job, however threads also have some of their own properties, such as a call stack. The job's properties contain information about how the work is processed. The job serves as the owner for properties that are shared among threads within the same job. Work management provides a way for you to control the work done on your system through a job's properties.

The general properties of a job determine how the system runs each job. Some of the properties are grouped together in the job description for easier multiple job management. The system knows what properties to get and when, based on how the job properties are specified. The iSeries system runs different types of jobs to serve various needs. Most job types use a job description.

For more information about jobs, see the following topics:

Active and inactive jobs
Learn what active and inactive jobs are.

Job types
Learn about the different types of jobs that run on the iSeries.

Job properties
Learn how to work with job properties.

Job actions
Learn how to manage jobs through iSeries Navigator.

Threads
Learn the difference between threads and jobs.

Job queues
Learn how a job goes from waiting on the job queue to performing work.

A job's life
Learn what happens during a job's life from the start to the end.

Note: APIs, such as Open List of Jobs (QGYOLJOB) and Retrieve Job Information (QUSRJOBI), can be called to get information on jobs. For more information on APIs, see Application programming interfaces (APIs).

Active and inactive jobs

Active jobs:

Active jobs are jobs that have started running but have not completed running. Following are some characteristics of an active job:

- Contains running code
- Has a call stack
- Has objects locked

- Has the status of an active job, for example:
  Running
  Waiting for (x)

For information about the properties of active jobs, see Job properties.

To learn how to manage active jobs, see Manage jobs and threads.

Inactive jobs:

Inactive jobs are jobs on a job queue waiting to be started, or jobs that have completed processing (ended) but are waiting for a printer output file (also called spooled files) to be printed.

Job types

The iSeries server processes several different job types. You can select one of the following job types to learn more about that job type.

Server jobs are jobs that have set the server type using the Change Job (QWTCHGJB) API, and they will have an additional classification of Server with one of the following job types:

Autostart
An autostart job is started automatically when the subsystem it is associated with starts.

Batch
A batch job is a predefined group of processing actions that is submitted to the system.

Communications
A communications job is a batch job that was started by a program start request from a remote system.

Interactive
An interactive job requires input from a signed-on user and an iSeries server.

Prestart
A prestart job is a batch job that starts before a work request is received. The two types of prestart jobs:

- Prestart communications - The job is a communications batch job that starts running before a remote system sends a program start request.
- Prestart batch - The job is a batch job that starts before a work request is received.

Reader and writer
A reader job is a spooled input job, and a writer job is a spooled output job.

Subsystem
The subsystem job provides control over an active subsystem.

System
System jobs are created by the operating system to control system resources and perform system functions.

Autostart jobs: An autostart job starts automatically when the subsystem it is associated with starts. These jobs generally perform initialization work that is associated with a particular subsystem. Autostart jobs can also perform repetitive work or provide centralized service functions for other jobs in the same subsystem.

The subsystem job uses information from the autostart job entry in the subsystem description, when starting a job.

Note: All autostart jobs are started when the subsystem starts. The value specified for the maximum number of jobs in the subsystem does not prevent the autostart jobs from starting. If the maximum number of jobs in the subsystem is exceeded, no other jobs can be started. When enough autostart jobs have completed so that the number of jobs running is below the maximum activity level, other jobs in the subsystem can start.

For more information about autostart jobs and how they start, see the Autostart Jobs (Chapter 9) and Autostart Job Entry (Chapter 4) topics in the Work Management manual

.

Batch jobs: A batch job is a predefined group of processing actions that is submitted to the system. Batch jobs run in the system background, freeing the user who submitted the job to do other work. The job requires no interaction on the part of the user once it has been set up. Batch jobs are typically low priority jobs. Several batch jobs can be active at the same time.

Following are different kinds of batch jobs:

Simple batch job
Most people are familiar with the simple batch job that is submitted to a job queue. For more information about a simple batch job's life, see A job's life.

Batch immediate job
A batch immediate job is a batch job that was started with many of the attributes of its parent job. The job runs in the same subsystem as the parent job. Because the job copies attributes from the parent job and does not go through a job queue, it can start faster than jobs submitted to a job queue.

Batch MRT job
A batch MRT job is a multiple requester terminal (MRT) job. MRT jobs are S/36 Environment jobs that act like servers, allowing other S/36 Environment jobs to attach to them in order to run an MRT procedure.

Batch print job
Batch print jobs track the printer output files (also called spooled files) that were created by a job whose current user profile is different from the user profile that it was started under.

For more information, see How a Batch Job Starts in Chapter 8 of the Work Management

manual.

Communications jobs: Communication jobs are started when a program start request is received from a remote system. For performance reasons, instead of starting a communications job each time a program start request is received, you can configure a prestart job to handle a program start request from a remote system.

For more information about a program start request, see chapter 3 of the ICF Programming

manual.

For more information, see Communications Jobs in Chapter 10 of the Work Management

manual.

Interactive jobs: Interactive jobs require continual two-way communications between the user and the iSeries server to perform a task. An interactive job begins when a user signs onto a system. The system requests sign-on information. If the sign-on request is accepted by the system, then the system creates the interactive job. The system then asks the user to supply a request. The user enters a request, and the system responds by processing the request. This pattern is repeated until the user ends the interactive job by signing off the system. If an interactive job is part of a group of jobs or a pair of jobs, then it will have one of the following job types:

Interactive - Group
An Interactive - Group job is part of a group of jobs that is associated with a single display device.
Interactive - System request
An Interactive - System request job is one of a pair of jobs that is associated with each other by the system request function.

Prestart jobs: A prestart job starts before a work request is received, either when the subsystem starts or as a result of the Start Prestart Jobs (STRPJ) command. Prestart jobs start from a prestart job entry (PJE) in the subsystem description. The prestart job entry specifies properties such as what program to run in the prestart job, the user profile under which the prestart job starts running, the job description, the class used to specify the run-time properties of the job, and the memory pool in which the prestart job runs.

Prestart jobs can start and initialize themselves before a work request is received. This reduces the amount of time required to handle the requests. A new job is not required for every work request. In addition, prestart jobs provide the ability to initialize once and handle many requests so that a new job is not needed for every request. Most client server applications use prestart jobs to handle the requests for the client user. Having a job ready to go makes the performance better in this situation because the prestart job can start processing the request for the user immediately.

Note: The value specified for the maximum number of jobs in the subsystem can prevent prestart jobs from starting. If the maximum number of jobs in the subsystem is exceeded, no prestart jobs can be started. When enough jobs have completed so that the number of jobs running is below the maximum number of jobs in the subsystem, prestart jobs in the subsystem can start.

Two types of prestart jobs exist. Each type handles different types of requests. Before a job waits for its first request, it will be shown as Prestart only because the system does not know yet what type of requests the job will handle. Following are the two types of prestart jobs:

Prestart communications job
A prestart communications job is a communications batch job that starts running before a remote system sends a program start request.

For more information about prestart communications jobs, see Prestart Jobs in Chapter 11 of the Work Management

manual.

Prestart batch job
A prestart batch job is a batch job that starts before a work request is received.

Reader and writer jobs: **Reader**
A reader job reads batch job streams from database and diskette files, and places the jobs on a job queue. The reader job is part of input spooling and is an IBM-supplied program.

Writer
A writer job writes records from printer output files (also called spooled files) to a printer. The writer job is an IBM-supplied program, started in the spooling subsystem where it selects files from the output queue to be printed.

Subsystem jobs: A subsystem job (sometimes called subsystem monitor job) is created by the operating system to manage resources and to start, control, and end jobs. The subsystem job provides control over an active subsystem. Many subsystem jobs can run on a system at any time.

For more information, see Subsystems.

System jobs: System jobs are created by the operating system to control system resources and perform system functions. System jobs run when the iSeries server starts, without user input. These jobs perform a variety of tasks from starting the operating system, to starting and ending subsystems, to scheduling jobs.

Following are different kinds of system jobs and their functions:

System startup jobs: **Scpf (start control program function)**
This is the central job when you start the system. Scpf starts all system jobs except Qlus and brings the system to a usable state. This job remains active after the system starts, providing an environment for the running of low-priority and possibly long-running system functions. Scpf also runs during the power down (Pwrdwnsys) processing, and is the job that ends the machine processing.

Qwcbtclnup (job table cleanup)
This job is used during the start of the system to ensure that the job structures are available for use. It usually completes processing before the end of the system startup, but it can continue running after the system starts, if there are a lot of job structures to clean up. This system job ends when it completes processing.

System arbiters: **Qsysarb (system arbiter)**
The system arbiter provides the environment for the running of high-priority functions. It handles system resources and keeps track of the state of the system. The system arbiter responds to system-wide events that must be handled immediately and those that can be handled more efficiently by a single job. Qsysarb and Qcmnarbxx (communications arbiters) are responsible for processing communication requests, device locking, line, controller, and device configuration, and handling of other system-wide resources.

Qsysarb2 (system arbiter 2)
This job is responsible for managing tape resources, handling command analyzer spaces for command processing and other system-wide processing for the operating system.

Qsysarb3 (system arbiter 3)
This job is responsible for creating and maintaining the job structures on the system. Whenever temporary or permanent job structures are required for job initiation, the request is processed by Qsysarb3.

Qsysarb4 (system arbiter 4)
This job is responsible for starting and ending subsystems. This includes the initial power down (Pwrdwnsys) processing.

Qsysarb5 (system arbiter 5)
This job is responsible for processing machine events. This includes handling events to support auxiliary power, continuous powered mainstore (CPM), system auxiliary storage pools (ASPs) and storage threshold, and lock table limits. Usually, the machine events are handled and corresponding CPF messages are sent to Qsysopr and Qhst.

Communications jobs: **Qlus (logical unit services)**
Qlus handles the event handling for logical unit devices, known as communications devices. Qlus is also responsible for allocating devices to the correct communications subsystem.

Qcmnarbxx (communications arbiters)
The communications arbiters along with Qsysarb (system arbiter) process work for all types of devices, not just communications devices. This work includes communications connection, disconnection, device locking, and error recovery processing. All device-related work is spread throughout the Qcmnarbxx jobs and the system arbiter.

The Qcmnarbxx system value determines the number of communications arbiter jobs that are started. A minimum of three communications arbiters are started on single-processor systems.

Qsyscomm1 (system communications)
This job handles some communications and input/output (I/O) activity.

Q400filsvr (remote file system communication)
This job performs the common programming interface communications (APPN or APPC) for the remote file system.

Database jobs: **Qdbfstccol (database file statistic collection)**
This job collects database file statistics. These statistics are crucial to proper database query optimization.

Qdbsrvxr (database cross-reference)
This job maintains each of the file level system cross-reference files in Qsys. These files contain cross-reference information about database files and SQL information across the system. The files all begin with the prefix of Qadb in library Qsys. The primary file that must be maintained is Qadbxref, the cross-reference file. This file contains a record of each physical database, logical database, DDM, and Alias file on the system. Qdbsrvxr activates when a file is created, changed, deleted, restored, renamed, or its ownership is changed.

Qdbsrvxr2 (database cross-reference 2)
This job maintains the two field level cross-reference files. Qadbifld in library Qsys is the field cross-reference file. Qadbkfld in library Qsys is the key field cross-reference file. Qdbsrvxr2 is activated when a file is created, changed or deleted.

Qdbsrv01 (database server)
This job can be viewed as the database maintenance task dispatcher. The number of database server jobs on the system is one plus twice the number of processors, or one plus twice the number of ASPs, whichever is greater. The minimum started is five. Qsbsrv01 is the main system job assigning work to the others. Typically, Qdbsrv01 will be most active immediately after restoring a library that contains database files. Its function includes:
- Signaling to the system-managed access path protection (SMAPP) Licensed Internal Code (LIC) tasks that new access paths have been restored. SMAPP then determines whether these access paths need to be protected.
- Preparing the list of access paths that are required to be rebuilt because the access paths were not restored.

Of the remaining database server jobs, the first half process high-priority requests, and the second half process low-priority requests. Qdbsrv02 through Qdbsrv05 are high priority, Qdbsrv06 through Qdbsrv09 are low priority.

Qdbsrvxx (database server, high priority)
These jobs perform journal and commitment control maintenance for the system and are considered quick or short-running work.

Qdbsrvxx (database server, low priority)
These jobs perform access path maintenance on user data files. Typically, these jobs are inactive, but in certain cases, they may activate to perform access path rebuilds. Some reasons why these jobs could be active are:

- Restoring database files that were not saved with access paths.
- Restoring logical files without the physical file they are based on.
- Canceling of an Rgzpfm command while in process.
- Invalidation of an index due to damage found in the index.
- Post-iSeries installation activity to complete cross-reference or other DBupgradede activity.
- Constraint verification

Qqqtemp1 and Qqqtemp2 (database parallelism)
The database parallelism system jobs perform asynchronous database processing for the DB2 Multisystem. If users query distributed files, the jobs are used to speed up the queries by doing certain tasks in parallel.

Other jobs: **Qalert (alert manager)**
This job performs the tasks necessary to process alerts (for information about alerts, see the Alerts Support

manual). This includes such activities as processing alerts received from other systems, processing locally created alerts, and maintaining the sphere of control.

Qdcpobjx (decompress system object)
These jobs decompress newly installed operating system objects as needed. There is a storage requirement for these jobs to run. If available storage on your system drops below a certain limit, these jobs will end. The number of decompress system object jobs is the number of processors plus one.

Qfilesys1 (file system)
This job supports the background processing of the integrated file system. It ensures that changes to files are written to storage and also performs several general file system cleanup activities.

Qjobscd (job schedule)
This job controls the system's job scheduling functions. Qjobscd monitors the timers for job schedule entries and scheduled jobs.

Qlur (LU 6.2 resynchronization)
Qlur handles the two-phase commit resynchronization processing.

Qpfradj (performance adjustment)
This job manages changes to the storage pool sizes and activity levels. All requests to change storage pools are processed by this job. In addition, if system value Qpfradj is set to a value of 2 or 3, this job dynamically changes the sizes and activity levels of storage pools to improve the system performance.

Qsplmaint (system spool maintenance)
This job performs system spooling functions.

Job properties

Job properties contain information about how jobs are processed. They are originally specified when the job is created. Some of the properties come from the job description. After the job is created, the job properties can be viewed and managed through Work Management in iSeries Navigator. The job properties pages in iSeries Navigator make a system operator's job easier by providing efficient and easy-to-use functions for managing jobs. Job properties can be viewed by any user, but can only be changed by a user with the proper authority. Similarly, an authorized user can manage jobs through job actions. Properties for system jobs cannot be changed in iSeries Navigator. However, the run priority of a job can be changed in the character based interface using the CHGSYSJOB command.

Work with job properties To view or change a job's properties, follow these steps:

1. In iSeries Navigator, expand **My Connections**.
2. In My Connections, expand the connection for your **iSeries server**.
3. Expand **Work Management**.
4. Double-click **Active Jobs** or **Server Jobs**, depending on the type of job you want to work with.
5. Find the job whose properties you want to view or change.
6. Right-click the **Job Name**.
7. Select **Properties**.

Job property sheets

General job properties allow you to view general information about jobs. This information includes the job's name and its job type, when the job entered the system, when the job started, the job's detailed status, and other information.

Performance properties allow you to view basic performance information and make changes that will affect a job's performance. You can view the performance statistics that have been calculated over the life of the job, such as CPU and disk I/O. You can change the following values that affect how the job will run:

- Run priority
- Time slice
- Default wait time

You can also view, refresh, set up an automatic refresh, or reset the Elapsed performance statistics that have been calculated for an active job. For more information, see Elapsed performance statistics.

Job Queue properties are available for jobs that are on a job queue or started from a job queue. You can change information for jobs currently on a job queue. You can work with the priority of the job on the job queue, view the date and time the job was placed on the job queue, and change when to make the job available to run.

Printer Output properties allow you to view and change properties that affect the printing of output for the job. You can also display the printer output for a job by using the printer output button. You can choose to

»

detach printer output

«

from a job, select a printer, choose the output queue and its library, specify the order that you want the information printed (priority), specify a page footer, and specify whether border and header information should be printed.

Messages properties allow you to specify how inquiry and break messages will be handled. If the job is a batch job, the message severity level that causes the job to end is also shown.

Job Log properties allow you to view and change information related to the job log as well as display the job log. The job log contains information that is related to requests entered for a job, such as commands in the job, commands from CL programs, and messages. This page allows you to specify whether or not to keep messages in the job log, what action the job needs to take when the job log is full, what kind of messages to keep, whether a printed job log (printer output) is generated for jobs that end normally, and the amount of detail to include for each message. For more information, see Job logs.

Security properties allow you to view security properties for jobs that are currently active. This includes the job user identity, the method used to set the job user identity (Set by), the current, user and the names of the group profiles that are associated with the initial thread of a job (Groups).

International properties allow you to view or change properties related to text and character format, and the language and country/region associated with the job. This includes the format to use when dates, times, and decimals are represented. There is also an indication of whether the job is capable of handling double-byte character sets (DBCS).

Threads properties allow you to view information related to threads for a job that is currently active or on a job queue. You can also display the threads for a job by using the Threads button. This page includes information about whether the job can run with multiple user threads, the number of active threads in the job, and the maximum number of user and system threads that the job can run with at any time.

Server properties allow you to view information about server jobs. For each server job, you can see the type of server, job user identity, and if available, the client IP address. The client IP address is the address of the user that this server is currently servicing.

Other properties allow you to view and change properties related the accounting code, switch settings, and whether or not to keep DDM connections active. You can also view the disk pool group, job date and whether the job is running in a System/36 special environment.

For more information, refer to the iSeries Navigator help.

Detach printer output: In releases prior to V5R2, printer output was attached to a job until it was deleted either as a result of being sent to the printer or explicitly by the user.

You have the option to detach printer output from the job when the job ends. Printer output that is detached from the job is not deleted from the system, but resides on the output queue. This allows the job to leave the system, which frees up the job structures to be used by another job.

Note: If you choose to detach printer output from the job, you will no longer be able to look at printer output by going through the job. You will need to look at the actual output queue where the output resides to see it.

Elapsed performance statistics: The Elapsed performance statistics page allows you to view performance statistics, for an active job or thread, that are calculated over the elapsed time. This is important when you are monitoring a job or thread and in detecting potential problems. These statistics include CPU, disk I/O, page fault rate, average response time, and interactive transactions.

Note:	The elapsed performance statistics for a thread do not include average response time and interactive transactions.

You can change the viewing options for these statistics by selecting one of the following buttons from the **Elapsed performance statistics** page:

- **Refresh Now**
  Refreshes the elapsed performance statistics and extends the time period that the statistics are calculated.
- **Timed Refresh**
  Allows you to set up automatic refreshes of the elapsed performance statistics. This can be used to monitor the performance information for a job.
- **Reset Statistics**
  Clears the elapsed performance statistics and resets the time period that the statistics are calculated.

Detailed status: The current status of a job is viewed from the **General** page in Job properties, under **Detailed status**. An example of a detailed status is:

> **Scheduled to run at**
> The job remains waiting on the job queue until the scheduled date and time. At the scheduled time on the scheduled date, the job is available to be selected from the job queue.
>
> The detailed status can display an associated status value (status - x), which provides additional details about the current status of the job. An example of a detailed status plus the associated status value is: **Ended - CPU limit exceeded**
> *Ended* refers to the status of the job (the job has ended), and *CPU limit exceeded* represents why the job has that status (Ended).
>
> The detailed status can also have another associated status value displayed [status - x (x)] to reflect the current status of the job. For example a job that is ending could have the following status: **Ending - CPU limit exceeded (Waiting for lock)**
> The job is in the process of ending (Ending) because the CPU limit was exceeded (CPU limit exceeded), and the job is currently waiting for a lock (Waiting for lock) in the ending process.
>
> If the job does not end in a timely manner, this information can assist with problem analysis.

Status values can have additional information in the properties pages. For example, the status waiting for a lock, on the properties page, will show you what object is associated with the lock request.

End jobs: The two ways to end a job either controlled or immediate. Selecting **controlled** is usually the better choice because it allows programs running in the job to perform their end-of-job cleanup and to end properly. Selecting **immediate** ends the job immediately. It is recommended that immediately ending a job be done only after the controlled option has failed. Because the programs running in the job will not perform the normal application cleanup procedures when immediate is selected, you can get undesirable results such as application data that has been partially updated. iSeries Navigator allows you to specify a time limit for a controlled ending so that if it takes longer than the time you specify, an immediate ending will take place.

A job can check the end status for a job through the Job APIs such as the Retrieve Job Information (QUSRJOBI) API. When a controlled end is selected, an application that needs to perform end-of-job cleanup should detect the controlled end. One way an application can do this is by the asynchronous signal SIGTERM. When a job being ended in a controlled manner has a signal handling procedure for the asynchronous signal SIGTERM, the SIGTERM signal is generated for that job. When the signal handling procedure for the SIGTERM signal is given control, the procedure can take the appropriate actions to allow the application to be ended in a controller manner.

For detailed steps about how to end a job, see Ending a job.

For more information about ending a job and detecting the controlled end, see **Ending a Job** in chapter 5 of the Work Management

manual.

Details: Active job actions: The Details menu in the Work Management folder provides access to the following resources that are being used by the job or initial thread of the job:

Call stack

The call stack for the job is displayed. The call stack is the programs and procedures that are being used. This is helpful for finding out what program a job is running and what the job is doing.

Library list

The library list for the selected job or thread is displayed. A library list is a list of system and user-created libraries to search and the order that they are to be searched. A library is a container for objects, and all objects on the iSeries server require a reference consisting of the object name and a library. It is important to have the library list properly established because objects are found by searching the libraries. If the library list is not properly established, the job may not find an object or it may find the object in the wrong library. IBM supplies some libraries (library names that begin with **Q**), but you can also create your own. By selecting a library from this dialog and right-clicking, you can work with the properties of that library.

Locked objects

The list of

»

locked objects

«

and the objects for which the job or thread is waiting for a lock are displayed. This allows you to see what objects a job is using as well as the objects the job is attempting to use.

Open files

A list of open files and details of how that file has been used, such as how many I/O operations have occurred for the job selected, are displayed. Viewing this list is helpful for debugging and for checking the status of a job.

»

Threads

A list of threads running within a job. The initial thread, by default, is listed at the top of the window. Threads are independent pieces of work that help the job process more than one thing at a time.

«

»
Transactions

A list of transactions associated with the job. A transaction is a logical unit of work on the iSeries system. It is commonly referred to in relation to database operations. For more information on Transactions, see the iSeries Navigator help or go to Transactions.

«

Elapsed performance statistics

A list of elapsed performance statistics calculated over a period of time is displayed. This information is helpful for monitoring jobs and can assist with problem analysis.

»
Last SQL statement

The Last SQL statement option displays the last SQL statement run in a job. This SQL statement is displayed in Run SQL Scripts. From Run SQL Scripts, you can re-run the statement, edit and run the statement, or save the statement to a database file or PC file.

«

Job logs: The job log displays a list of messages that are associated with a specific job. Additional information about the messages, for example the date and time they were sent, is also displayed. Because the date and times are recorded in the job log, you can determine when an error occurred. By selecting **Details** from the **File** option on the menu bar, more information about the message can be displayed, such as the cause of the message and an explanation of what action should be taken, if any, to recover from the error. For job log messages, you can click the Advanced button to see information about the program that sent the message and the program to which the message was sent. You can make changes to how the job log is handled and what information is logged in the job log on the **Job Log** page in the Job Properties dialog.

For information about how to view the job log for jobs, see Accessing job logs.

Threads

A thread is an independent unit of work within a job that uses many of the jobs resources to complete work. The difference between jobs and threads is that threads run within the job helping it to finish its work. Every active job has at least one thread, which is called an initial thread. The initial thread is created as part of starting the job. The use of threads within a job allows many things to be done at once. For example, while a job is processing, a thread may retrieve and calculate data needed by the job to finish processing.

For more information on threads, see the following topics:

- **Thread actions**
  Manage threads through iSeries Navigator.
- **Thread types**
  This covers the different types of threads running within a job.
- **Thread status**
  This includes the different statuses of a thread.

«

Thread actions: Threads help jobs process more than one operation at a time while running. Monitoring the threads that are running within a job may be necessary as you attempt to keep the job running efficiently. Once you find the thread you want to manage, the following actions are available by right-clicking the thread.

Reset Statistics
Allows you to reset the list information you are viewing, and it sets the elapsed time to 00:00:00.

Details
Because the functions of a thread are similar to that of a job, they share some of the same job actions. Details contains detailed information about the following thread actions:
- Call stack
- Library list
- Locked Objects
- Transactions
- Elapsed Performance Statistics

Hold
Allows you to hold the thread. Threads can be held multiple times. The operating system keeps track of the number of times a thread is held.

Release
Releases the thread that was held. The thread must be released each time that it is held in order for it to run.

Delete/End
Allows you to end the selected thread or threads. For more information, see Ending a thread.

Thread Properties
Displays the different properties of a thread.

For more detailed information on the actions you can perform on Threads, see the iSeries Navigator help.

«

Thread types: The thread type determines how the thread was created on the system.

The types of threads are:

User
The thread is created by the customer application. The initial thread in a job is always a user thread. The Allow multple threads field must be must be set to yes for multiple user threads to be used.

System
The thread is created by the system on behalf of the user. Some system functions use system threads to complete processing. If a customer's application uses a system function that uses threads, system threads are used.

Note: In the threads on iSeries Navigator, by default, you will see **Initial** as the type of the first thread in the list. The initial thread is the first thread created within the job when it starts. In iSeries Navigator, the initial thread is represented by this

icon. You can never delete or end the initial thread.

Thread status: The current status of a thread is viewed from the **General** page in the Thread properties dialog, under Detailed status. An example of a detailed status is:

Waiting for dequeue

The thread of the job is waiting for completion of a dequeue operation. A dequeue is an operation for removing messages from queues. Messages are communications sent from one person or program to another. In particular, a message is enqueued (placed) on a queue system object by one thread and dequeued (removed) by another thread.

Note: When Waiting for dequeue is shown on a properties page, additional information that identifies the queue being waited on is displayed. When the job or thread is waiting on the dequeue operation to complete for an OS/400 object, you will see a 10-character object name, its library, and the object type. If the job or thread is waiting on the dequeue operation to complete for an internal object, you will see a 30-character object name. For internal objects you need job control special authority (*JOBCTL) to see the 30-character name.

The detailed status can display an associated status value (status - x), which provides additional details about the current status of the thread. An example of a detailed status plus the associated status value is:

Held (n)

An individual thread is held. Unlike a job, a thread can have multiple holds on it at the same time. A number (for example, Held (3)) following the thread status tells the user how many times that thread has been held without being released. For example, if a thread has had three holds put on it and then has been released once, it still has two holds against it. A number is only shown when the status appears on the Properties page and will not appear when displayed in a list. To resume thread processing, select the Release action for the thread.

For more information on the different thread statuses, see the iSeries Navigator help.

Job queues

A job queue contains an ordered list of jobs waiting to be processed by a subsystem. The job queue is the first place that a submitted batch job goes before becoming active in a subsystem. The job is held here until a number of factors are met. In order for jobs on a job queue to be processed, there must be an active subsystem that is accepting work from that job queue. When a subsystem starts, it attempts to allocate the job queues that it is configured to accept work from, and it must successfully allocate a job queue in order to process jobs from that job queue. Therefore, while one subsystem may be processing jobs from multiple job queues, only one subsystem may be processing jobs from a particular job queue at a time.

Subsystems select jobs from job queues in priority order, within limits that may be configured for each priority. Each job has a job queue priority that can be managed when the job is on the job queue through job properties. A base set of job queues is provided with your system. In addition, you may create additional job queues that you need.

Note: APIs, such as Open List of Job Queues (QSPOLJBQ) and Retrieve Job Queue Information (QSPRJOBQ), can be called to get information on job queues. For more information on APIs, see Application programming interfaces (APIs).

For more information about jobs on job queues, see the following topics:

- **How work enters the system**.
  Understand how work gets onto a job queue.
- **How a job queue works**
  Understand how a job gets from a job queue to a subsystem.
- **Creating a job queue**
  Create a job queue with information in Chapter 8 of the Work Management

  manual.

How a job queue works

Jobs are taken from a job queue to do work in a subsystem after the job queue is allocated by an active subsystem. The different factors that determine how the jobs are selected from a job queue. Jobs that are not coming off of a job queue can be moved from one job queue to another, in order for better efficiency.

The following determine how jobs are taken from a job queue:

Maximum active jobs for subsystems
This represents the maximum number of jobs that can be running in a subsystem. Once this limit is reached, no more jobs can start in the subsystem.

Maximum active jobs for job queues
This represents the maximum number of jobs from the job queue that can be running in a subsystem at the same time. Once this limit is reached, no more jobs can start from that job queue.

Priority on job queue
Jobs that are waiting to run are selected based on the job queue priority. The subsystem attempts to run higher priority jobs first (job queue priority ranges from 0 through 9 where 0 is the higher priority), but if the number of jobs running from a priority level reaches the Maximum Active Jobs value per priority level, the next priority level is processed. (If jobs with the same priority enter the job queue, the first job submitted will run first, then the second, and so on.)

For detailed information, see Change the priority of a job within a job queue.

Sequence
You specify the sequence in the job queue entry of the subsystem description. The sequence number defines the order in which the subsystem will process the job queues. The subsystem takes jobs from the job queue with the lowest sequence number first. If there are no more jobs on the job queue, or if one of the maximum values associated with the job queue is reached, the subsystem will process the job queue with the next highest sequence number.

For detailed information about moving jobs, see Move jobs to different job queues.

Subsystems

The **subsystem** is where work is processed on the iSeries server. All jobs, with the exception of system jobs, run within subsystems.

More technically, a subsystem is a single, predefined operating environment through which the system coordinates work flow and resource use. The system can contain several subsystems, all operating independently of each other. Subsystems manage resources. Each subsystem can run unique operations. For instance, one subsystem may be set up to handle only interactive jobs, while another subsystem handles only batch jobs. Subsystems can also be designed to handle many types of work. The system allows you to decide the number of subsystems and what types of work each subsystem will handle.

A subsystem can be either active or inactive. An active subsystem is one that has been started (see how subsystems start for details). An inactive subsystem is one that either has not yet been started, or has been stopped (see how subsystems stop for details).

The **controlling subsystem** is the interactive subsystem that starts automatically when the system starts, and it is the subsystem through which the system operator controls the system during system startup.

A **subsystem job** is a job created by the operating system to manage resources and to start, control, and end jobs.

Note: APIs, such as Retrieve Subsystem Information (QWDRSBSD) and Retrieve System Status (QWCRSSTS), can be called to get information on subsystems. For more information on APIs, see Application programming interfaces (APIs).

See the following for more information on subsystems:

Subsystem description
The run-time characteristics of a subsystem are defined in the subsystem description.
Subsystems shipped with the system
Two complete subsystem configurations are supplied by IBM.
User-defined subsystems
You can create your own subsystem description.
Subsystem properties
The attributes of a subsystem are provided.
Subsystem life cycle
This explains how work is processed on the iSeries server.

Subsystem description

The run-time characteristics of a subsystem are defined in an object called a **subsystem description**. A subsystem description acts as a set of instructions, telling the subsystem how, where, and how much work enters a subsystem, and which resources the subsystem uses to perform the work. A subsystem is created when a subsystem description is defined or created. An active subsystem takes on the simple name of the subsystem description.

For details on what information is contained in the subsystem description, see the following table:

Information in subsystem description		**Description**	**Additional information (Work Management manual)**
Subsystem attributes		Specifies overall system characteristics: • Operational attributes such as the number of jobs that can be active in the subsystem at the same time, and the sign-on display. • Memory pools used by the subsystem. • Authority to the subsystem description. • Text description of the subsystem description.	**Changing the sign-on display file**, Chapter 4 of the Work Management manual.
Work entries		The work entry in a subsystem description specifies the source from which jobs can be accepted for processing in the subsystem. In other words, the location where work can enter the subsystem.	Work entries, Chapter 4 of the Work Management manual.
	Autostart job entry	Identifies the autostart jobs to start as soon as the subsystem starts.	Autostart jobs, Chapter 9 of the Work Management manual.
	Communications entry	Identifies the communications device that another system uses to submit work.	Communications jobs, Chapter 10 of the Work Management manual.
	Job queue entry	Identifies the job queue from which to take work and determine how much work to accept.	Batch jobs, Chapter 8 of the Work Management manual.
	Prestart job entry	Identifies the information used when prestart jobs are started.	Prestart jobs, Chapter 11 of the Work Management manual.
	Workstation entry	Identifies the workstation from which to take work.	Interactive jobs, Chapter 6 of the Work Management manual.

Information in subsystem description	Description	Additional information (Work Management manual)
Routing entries	Identifies the subsystem memory pool to use, the controlling program to run, and run-time information.	Routing entries, Chapter 4 of the Work Management manual.

Subsystem Description objects are shipped with every system. Below are the updates to the shipped subsystem descriptions on the iSeries server. For each object, this table provides:

Object Name
Object description
Command used to create the object
Object parameters other than the default

The objects are grouped by object type.

This table and Appendix C in the Work Management manual

will give you allow you to see most of the shipped subsystem descriptions on the iSeries.

Object	Addition, Deletion, or Update	Parameters other than default
QBASE	Added a communication entry (ADDCMNE)	SBSD (QSYS/QBASE) DEV (Q1PLOC) DFTUSR (*NONE) MODE (Q1PMOD) MAXACT (0)
QBASE	Added a communication entry (ADDCMNE)	SBSD (QSYS/QBASE) REMLOCNAME (Q1PLOC) DFTUSR (*NONE) MODE (Q1PMOD) MAXACT (0)
QBASE	Added prestart job entry (ADDPJE)	SBSD (QSYS/QBASE) PGM (QSYS/QZSCSRVR) USER (QUSER) STRJOBS (*YES) INLJOBS(1) THRESHOLD (1) ADLJOBS(3) JOB (*PGM) JOBD (*USRPRF) MAXUSE (1) WAIT (*YES) POOLID (2) CLS (QGPL/QCASERVR *CALC *NONE *CALC)

Object	Addition, Deletion, or Update	Parameters other than default
QBASE	Added prestart job entry (ADDPJE)	SBSD (QSYS/QBASE) PGM (QSYS/QNPSERVR) USER (QUSER) STRJOBS (*YES) INLJOBS(1) THRESHOLD (1) ADLJOBS(3) JOB (*PGM) JOBD (*USRPRF) MAXUSE (200) WAIT (*YES) POOLID (1) CLS (QGPL/QCASERVR *CALC *NONE *CALC)
QBASE	Added prestart job entry (ADDPJE)	SBSD (QSYS/QBASE) PGM (QSYS/QZRCSRVR) USER (QUSER) STRJOBS (*YES) INLJOBS(1) THRESHOLD (1) ADLJOBS(3) JOB (*PGM) JOBD (*USRPRF) MAXUSE (1) WAIT (*YES) POOLID (2) CLS (QGPL/QCASERVR *CALC *NONE *CALC)
QCMN	Added a communication entry (ADDCMNE)	SBSD (QSYS/QCMN) REMLOCNAME (Q1PLOC) DFTUSR (*NONE) MODE (Q1PMOD) MAXACT (0)
QCMN	Added a communication entry (ADDCMNE)	SBSD (QSYS/QCMN) DEV (Q1PLOC) DFTUSR (*NONE) MODE (Q1PMOD) MAXACT (0)
QCMN	Added prestart job entry (ADDPJE)	SBSD (QSYS/QCMN) PGM (QSYS/QZRCSRVR) USER (QUSER) STRJOBS (*YES) INLJOBS(1) THRESHOLD (1) ADLJOBS(3) JOB (*PGM) JOBD (*USRPRF) MAXUSE (1) WAIT (*YES) POOLID (1) CLS (QGPL/QCASERVR *CALC *NONE *CALC)

Object	Addition, Deletion, or Update	Parameters other than default
QCMN	Added prestart job entry (ADDPJE)	SBSD (QSYS/QCMN) PGM (QSYS/QZSCSRVR) USER (QUSER) STRJOBS (*YES) INLJOBS(1) THRESHOLD (1) ADLJOBS(3) JOB (*PGM) JOBD (*USRPRF) MAXUSE (1) WAIT (*YES) POOLID (1) CLS (QGPL/QCASERVR *CALC *NONE *CALC)
QCMN	Added prestart job entry (ADDPJE)	SBSD (QSYS/QCMN) PGM (QSYS/QNPSERVR) USER (QUSER) STRJOBS (*YES) INLJOBS(1) THRESHOLD (1) ADLJOBS(3) JOB (*PGM) JOBD (*USRPRF) MAXUSE (200) WAIT (*YES) POOLID (1) CLS (QGPL/QCASERVR *CALC *NONE *CALC)
QSERVER	Added prestart job entry (ADDPJE)	SBSD (QSYS/QSERVER) PGM (QSYS/QZDAINIT) USER (QUSER) STRJOBS (*YES) INLJOBS(1) THRESHOLD (1) ADLJOBS(3) JOB (*PGM) JOBD (*USRPRF) MAXUSE (1) WAIT (*YES) POOLID (1) CLS (QGPL/QPWSERVER *CALC *NONE *CALC)
QSERVER	Added prestart job entry (ADDPJE)	SBSD (QSYS/QSERVER) PGM (QSYS/QPWFSERVSO) USER (QUSER) STRJOBS (*NO) INLJOBS(1) THRESHOLD (1) ADLJOBS(2) MAXJOBS (*NOMAX) JOBD (*USRPRF) JOB (*PGM) MAXUSE (200) WAIT (*YES) POOLID (1) CLS (QGPL/QPWFSERVER *CALC *NONE *CALC)

Object	Addition, Deletion, or Update	Parameters other than default
QSYSWRK	Added job queue entry (ADDJOBQE)	SBSD (QSYS/QSYSWRK) JOBQ (QSYS/Q1PSCHQ) MAXACT (1) SEQNBR (70)
QSYSWRK	Added job queue entry (ADDJOBQE)	SBSD (QSYS/QSYSWRK) JOBQ (QSYS/Q1PSCHQ2) MAXACT (1) SEQNBR (80)
QSYSWRK	Added job queue entry (ADDJOBQE)	SBSD (QSYS/QSYSWRK) JOBQ (QSYS/Q1PSCHQ3) MAXACT (1) SEQNBR (90)
QSYSWRK	Added an autostart job entry (ADDAJE)	SBSD (QSYS/QSYSWRK) JOB (QGLDPUBA) JOBD(QSYS/QGLDPUBA)
QSYSWRK	Added an autostart job entry (ADDAJE)	SBSD (QSYS/QSYSWRK) JOB (QGLDPUBE) JOBD(QSYS/QGLDPUBE)
QSYSWRK	Added autostart job entry (ADDAJE)	SBSD (QSYS/QSYSWRK) JOB (QPM400) JOBD (QSYS/Q1PJOBD)
QSYSWRK	Added a communication entry (ADDCMNE)	SBSD (QSYS/QSYSWRK) DEV (Q1PDEV) JOBD (*USRPRF) DFTUSR (QUSER) MODE (Q1PMOD) MAXACT (*NOMAX)
QSYSWRK	Added a communication entry (ADDCMNE)	SBSD (QSYS/QSYSWRK) DEV (Q1PLOC) JOBD (*USRPRF) DFTUSR (QPM400) MODE (Q1PMOD) MAXACT (*NOMAX)
QSYSWRK	Added a communication entry (ADDCMNE)	SBSD (QSYS/QSYSWRK) RMTLOCNAME (Q1PLOC) JOBD (*USRPRF) DFTUSR (QPM400) MODE (Q1PMOD) MAXACT (*NOMAX)
QSYSWRK	Added routing entries (ADDRTGE)	SBSD (QSYS/QSYSWRK) SEQNBR (2150) CMPVAL (TOTNTP) PGM (QSYS/QTOTSNTP) CLS (QSYS/QSYSCLS10)
QSYSWRK	Added routing entry (ADDRTE)	SBSD (QSYSWRK) SEQNBR (300) CMPVAL (PGMEVOKE 29) PGM (*RTGDTA) CLS (QSYS/QSYSCLS50) MAXACT (*NOMAX) POOLID (1)

Object	Addition, Deletion, or Update	Parameters other than default
QSYSWRK	Added routing entry (ADDRTGE)	SBSD (QSYS/QSYSWRK) SEQNBR (2536) CMPVAL ('QZSCSRVSD') PGM (QSYS/QZSCSRVSD) CLS (QGPL/QCASERVR)
QSYSWRK	Added routing entry (ADDRTGE)	SBSD (QSYS/QSYSWRK) SEQNBR (2537) CMPVAL ('QZHQSRVD') PGM (QSYS/QZHQSRVSD) CLS (QGPL/QCASESERVR)
QSYSWRK	Added routing entry (ADDRTGE)	SBSD (QSYS/QSYSWRK) SEQNBR (2538) CMPVAL ('QNPSERVD') PGM (QSYS/QNPSERVD) CLS (QGPL/QCASESERVR)
QSYSWRK	Added routing entry (ADDRTGE)	SBSD (QSYS/QSYSWRK) SEQNBR (2539) CMPVAL ('QZRCSRVSD') PGM (QSYS/QZRCSRVSD) CLS (QGPL/QCASESERVR)
QSYSWRK	Added routing entry (ADDRTGE)	SBSD (QSYS/QSYSWRK) SEQNBR (2540) CMPVAL ('QZSOSGND') PGM (QSYS/QZSOSGND) CLS (QGPL/QCASESERVR)
QSYSWRK	Added routing entry (ADDRTGE)	SBSD (QSYS/QSYSWRK) SEQNBR (2541) CMPVAL ('QZSOSMAPD') PGM (QSYS/QZSOSMAPD) CLS (QGPL/QCASESERVR)
QSYSWRK	Added routing entry (ADDRTGE)	SBSD (QSYS/QSYSWRK) SEQNBR (2170) CMPVAL ('QSYEIMMON') PGM (QSYS/QSYEIMMON) CLS (QSYS/QSYSCLS20) MAXACT (*NOMAX) POOLID (1)
QSYSWRK	Added routing entry (ADDRTGE)	SBSD (QSYS/QSYSWRK) SEQNBR (2200) CMPVAL ('QYASPPGM') PGM (QSYS/QYASPPGM) CLS (QSYS/QSYSCLS20) MAXACT (*NOMAX) POOLID (1)

Object	Addition, Deletion, or Update	Parameters other than default
QUSRWRK	Added prestart job entry (ADDPJE)	SBSD (QSYS/QSYSWRK) PGM (QSYS/QZSOSIGN) USER (QUSER) STRJOBS (*YES) INLJOBS(1) THRESHOLD (1) ADLJOBS(2) MAXJOBS (*NOMAX) JOB (*PGM) JOBD (QSYS/QZBSJOBD) MAXUSE (200) WAIT (*YES) POOLID (1) CLS (QGPL/QCASERVR *CALC *NONE *CALC)
QUSRWRK	Added prestart job entry (ADDPJE)	SBSD (QSYS/QUSRWRK) PGM (QSYS/QZSCSRVS) USER (QUSER) STRJOBS (*YES) INLJOBS(1) THRESHOLD (1) ADLJOBS(2) MAXJOBS (*NOMAX) JOB (*PGM) JOBD (QSYS/QZBSJOBD) MAXUSE (200) WAIT (*YES) POOLID (1) CLS (QGPL/QCASERVR *CALC *NONE *CALC)
QUSRWRK	Added prestart job entry (ADDPJE)	SBSD (QSYS/QUSRWRK) PGM (QSYS/QNPSERVS) USER (QUSER) STRJOBS (*YES) INLJOBS(1) THRESHOLD (1) ADLJOBS(2) MAXJOBS (*NOMAX) JOB (*PGM) JOBD (QSYS/QZBSJOBD) MAXUSE (200) WAIT (*YES) POOLID (1) CLS (QGPL/QCASERVR *CALC *NONE *CALC)

Object	Addition, Deletion, or Update	Parameters other than default
QUSRWRK	Added prestart job entry (ADDPJE)	SBSD (QSYS/QUSRWRK) PGM (QSYS/QZRCSRVS) USER (QUSER) STRJOBS (*YES) INLJOBS(1) THRESHOLD (1) ADLJOBS(2) MAXJOBS (*NOMAX) JOB (*PGM) JOBD (QSYS/QZBSJOBD) MAXUSE (1) WAIT (*YES) POOLID (1) CLS (QGPL/QCASERVR *CALC *NONE *CALC)
QUSRWRK	Added prestart job entry (ADDPJE)	SBSD (QSYS/QUSRWRK) PGM (QSYS/QZDASOINIT) USER (QUSER) STRJOBS (*YES) INLJOBS(1) THRESHOLD (1) ADLJOBS(2) MAXJOBS (*NOMAX) JOB (*PGM) JOBD (*USRPRF) MAXUSE (200) WAIT (*YES) POOLID (1) CLS (QGPL/QPWFSERVER *CALC *NONE *CALC)
QUSRWRK	Added prestart job entry (ADDPJE)	SBSD (QSYS/QUSRWRK) PGM (QSYS/QZHQSSRV) USER (QUSER) STRJOBS (*YES) INLJOBS(1) THRESHOLD (1) ADLJOBS(2) MAXJOBS (*NOMAX) JOB (*PGM) JOBD (QSYS/QZBSJOBD) MAXUSE (200) WAIT (*YES) POOLID (1) CLS (QGPL/QCASERVR *CALC *NONE *CALC)

Object	Addition, Deletion, or Update	Parameters other than default
QUSRWRK	Added prestart job entry (ADDPJE)	SBSD (QSYS/QUSRWRK) PGM (QSYS/QZDASSINIT) USER (QUSER) STRJOBS (*YES) INLJOBS(1) THRESHOLD (1) ADLJOBS(2) MAXJOBS (*NOMAX) JOB (*PGM) JOBD (QSYS/*USRPRF) MAXUSE (200) WAIT (*YES) POOLID (1) CLS (QSYS/QPWFSERVER *CALC *NONE *CALC)
QUSRWRK (moved from QSYSWRK TO QUSRWRK)	Added prestart job entry (ADDPJE)	SBSD (QSYS/QUSRWRK) PGM(QSYS/QRWTSRVR) USER (QUSER) STRJOBS (*YES) INLJOBS (1) THRESHOLD (1) ADLJOBS (2) MAXJOBS (*NOMAX) JOB (*PGM) JOBD (*USRPRF) MAXUSE (200) WAIT (*YES) POOLID (1) CLS (QSYS/QSYSCLS20 *CALC *NONE *CALC)

Subsystems shipped with the system

Two complete subsystem configurations are supplied by IBM and can be used without being changed.

The configuration the system uses when the system is started is controlled by the controlling subsystem description system value (QCTLSBSD). The default configuration consists of the following subsystem descriptions:

Qbase (controlling subsystem)		Qbase supports interactive, batch, and communications jobs. It has an autostart job, which automatically starts the Qusrwrk, Qserver, and Qspl subsystems.
	Qsyswrk	This is the system work subsystem. It contains jobs that support system functions that are started automatically at system startup and when the system comes out of restricted state.
	Qusrwrk	This is the user work subsystem. It contains jobs that are started by servers to do work on behalf of a user.
	Qserver	This is the file server subsystem.
	Qspl	This is the spool subsystem. It supports reader and writer jobs.

The other configuration, which is supplied by IBM, consists of the following subsystem descriptions:

Qctl (controlling subsystem)		Qctl has an autostart job, which automatically starts the Qinter, Qbatch, Qcmn, Qusrwrk, Qserver and Qspl subsystems.
	Qinter	This is the system work subsystem. It contains jobs that support system functions that are started automatically at system startup and when the system comes out of restricted state.
	Qbatch	This is the user work subsystem. It contains jobs that are started by servers to do work on behalf of a user.
	Qcmn	This is the file server subsystem.
	Qspl	This is the spool subsystem. It supports reader and writer jobs.
	Qsyswrk	This is the system work subsystem. It contains jobs that support system functions that are started automatically at system startup and when the system comes out of restricted state.
	Qusrwrk	This is the user work subsystem. It contains jobs that are started by servers to do work on behalf of a user.
	Qserver	This is the file server subsystem.

The Qbase configuration gives the ability to run all the same functions that you can run with the Qctl configuration and is easier to manage because it consists of fewer subsystems.

The Qctl default configuration allows for more individualized control over your system operations by dividing the system activity into different subsystems based on the type of activity. For example, if you want to run batch jobs over the weekend or overnight but do not want anyone to be able to sign on (except at the console), you can easily do that with the Qctl configuration by simply ending the Qinter subsystem.

If you are considering creating your own subsystem configuration, you may also find that it is easier to use the Qctl configuration as a starting point than the Qbase configuration.

User-defined subsystems

IBM provides subsystem descriptions that are shipped with the system. You can also create your own subsystem description. You can copy an existing subsystem description and change it, or you can create an entirely new description.

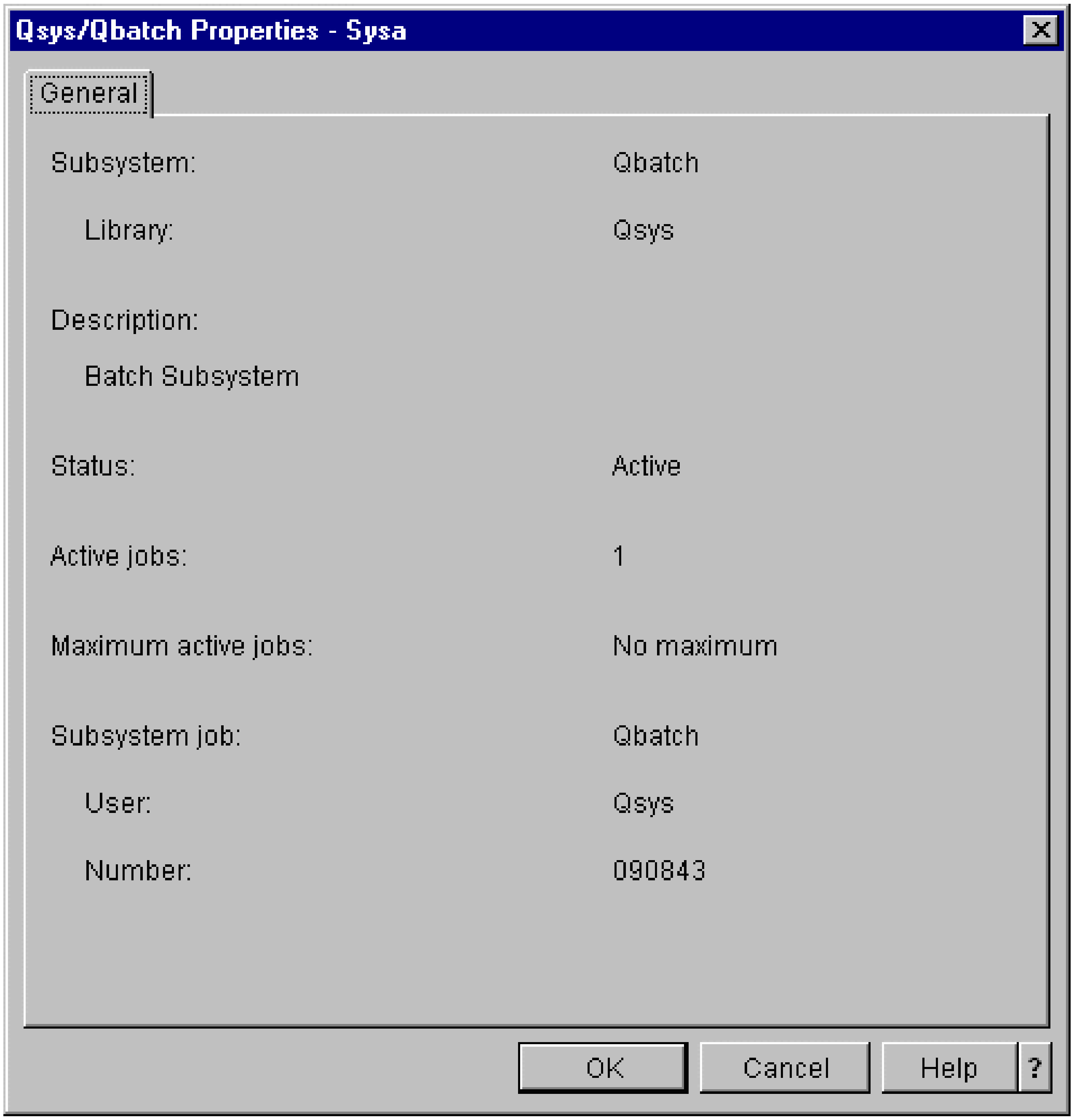
See Creating a subsystem description in Chapter 4 of the Work Management

manual for details.

Subsystem properties

Subystems have attributes, or properties. These properties give information about the current status of the subsystem, or about values identified in the subsystem description. Using iSeries Navigator, the following properties can be viewed for an active subsystem:

Subsystem	The name of the subsystem, as well as the library that contains the subsystem description.
Description	The description of the subsystem.
Status	The current status of the subsystem. The help contains details on the possible statuses.
Active jobs	The number of jobs currently active, either running or waiting to run, in the subsystem. This number does not include the subsystem job.
Maximum active jobs	The maximum number of jobs that can be active, either running or waiting to run, in the subsystem.
Subsystem job	The name of the subsystem job, including user and number.


Qsys/Qbatch Properties - Sysa

Subsystem:	Qbatch
Library:	Qsys
Description:	
Batch Subsystem	
Status:	Active
Active jobs:	1
Maximum active jobs:	No maximum
Subsystem job:	Qbatch
User:	Qsys
Number:	090843

To view the properties of a subsystem, follow these steps:

1. In iSeries Navigator, expand **My Connections**.
2. Expand the connection for your **iSeries server**.
3. Expand **Work Management**.
4. Expand **Subsystems**.
5. Expand **Active Subsystems**.
6. Right-click the subsystem you would like to view, then select **Properties**.

Subsystem life cycle

The life of a subsystem begins when it is started, and ends when the subsystem stops. In between, work is processed in the subsystem. See the following for details:

- Start a subsystem
- What happens when the subsystem starts
- Stop a subsystem

What happens when the subsystem starts: When a subsystem starts, the system allocates several items and starts autostart and prestart jobs before the subsystem is ready for work. The subsystem description is used to determine how items are allocated.

The following list represents the sequence of events that occur when the subsystem starts:

1. **Request to start subsystem is issued.**
2. **Memory pools are allocated.**
   Memory is allocated to the pools defined in the subsystem description. The memory that is allocated to each defined pool is taken from the Base memory pool. The system does not allocate memory to a pool if the amount of memory available to the Base memory pool would be less than the minimum size specified by the base memory pool minimum size (Qbaspool) system value. If the system cannot allocate all of the requested memory, it allocates as much memory as is available and allocates all the other as memory becomes available.
   See **Pool Allocation** in Chapter 4 of the Work Management

   manual.
3. **Display stations are allocated.**
   - If there are workstation entries and the device is varied on and has not been allocated by any other subsystem, the subsystem can allocate it and display the Sign-On display.
   - If the device is varied on and has been allocated by another subsystem and is at the Sign-On display (the Sign-On display was displayed before the second subsystem was started), a second subsystem can allocate the device from the first subsystem and display the Sign-On display.
   - If the device is not varied on, the subsystem cannot allocate it. The system arbiter (Qsysarb) and the Qcmnarbxx jobs hold locks on all varied-off devices.
   See **Workstation Device Allocation** in Chapter 4 of the Work Management

   manual.
4. **Communications devices are allocated.**
   Requests are sent to the Qlus (LU services) system job, which handles device allocation for all communications devices.
   See **Communications Devices** and **Mode Allocation** in the Work Management

   manual.
5. **Job queues are allocated.**
   The subsystem will not be able to allocate a job queue if it is already allocated to another active subsystem.
6. **Prestart jobs are started.**
7. **Autostart jobs are started.**
8. **Environment is ready for work.**

Memory pools

A **memory pool** is a logical division of main memory or storage that is reserved for processing a job or group of jobs. On the iSeries server, all main storage can be divided into logical allocations called memory pools. By default, the system manages memory pools. The system manages the transfer of data and programs into memory pools if necessary.

You can control how much work can be done in a subsystem by controlling the number and size of the memory pools. The greater the size of the memory pools in a subsystem, the more work that can be done in the subsystem.

Note: Although tuning and managing your system can help the efficiency of the flow of work through your iSeries server, it cannot account for inadequate hardware resources. Consider a hardware upgrade if the demands of your workload are significant..

The memory pool from which user jobs get their memory is always the same pool that limits their activity level. System jobs (such as Scpf, Qsysarb, and Qlus) get their memory from the base pool but use the machine pool activity level. Subsystem monitors get their memory from the first subsystem description pool but not the activity level. This allows a subsystem monitor to always be able to run regardless of the activity level setting.

Note: APIs, such as Retrieve System Status (QWCRSSTS), can be called to get information on memory pools. For more information, see Application programming interfaces (APIs)

See the following for more information on memory pools:
- Memory pool activity level
- Types of memory pools

Memory pool activity level

Memory pool activity levels allow for efficient use of system resource by limiting the number of threads that can be active at the same time in a memory pool.

The activity level of a memory pool is the number of threads that can actively use the CPU at the same time in a memory pool. The system manages the control of this level. Often during processing in a thread, a program waits for a system resource or a response from a workstation user. During such waits, a thread gives up its use of the memory pool activity level so that another thread that is ready to be processed can take its place.

When more threads are started than can run at the same time because of the activity level controls, the excess threads have to wait to use the processing unit (normally this wait is short). The memory pool activity level lets you limit the amount of main memory contention in the various memory pools in your subsystems.

The number of threads running (or active threads) refers to the number of threads that are eligible to compete for the processor and that count against the activity level for a memory pool. In this sense, active threads do not include threads that are waiting for input, for a message, for a device to be allocated, or for a file to be opened. Active threads do not include threads that are ineligible (threads that are ready to run but the memory pool activity level is at its maximum).

How activity levels work

More than one thread can be active at the same time in a memory pool because the processing for a thread can be briefly interrupted while needed data is retrieved from auxiliary storage. During this delay, which is usually short, another thread can run. Using the activity level, the machine can process a large number of threads in a memory pool and, at the same time, hold the level of contention to the limit you specify.

Maximum activity level

Once the maximum activity level for a memory pool has been reached, additional threads needing the memory pool are placed in the ineligible state to wait for the number of active threads in the memory pool to fall below the maximum activity level or for a thread to reach the end of its time slice. As soon as a thread gives up its use of the memory pool, the other threads that are not active become eligible to run by their priority. For example, if a running thread is waiting for a response from a workstation, it gives up its activity level and the activity level is no longer at its maximum.

Defining memory pool activity levels

Defining memory pools and activity levels correctly is generally dependent on size of the memory pool, the number of CPUs, the number of disk unit arms, and the characteristics of the application. See Performance tuning in Chapter 14 of the Work Management

manual for a more detailed description of how to set appropriate activity levels.

See Controlling levels of system activity in Chapter 4 of the Work Management

manual for more information.

Types of memory pools

A memory pool is a division of main storage or auxiliary storage. On the iSeries server, all main storage can be divided into logical allocations called memory pools. The two types of memory pools in a system are either private or shared. As many as 64 memory pools, in any combination of private and shared pools, can be active at the same time.

Private memory pool
Identified by subsystem name in iSeries Navigator, it is a pool in which a single subsystem can run jobs. Private pools are pools of main storage that cannot be shared by multiple subsystems. A private pool contains a specified amount of storage to be used by only one subsystem. You can have as many as 62 private pools allocated for use in active subsystems. A private pool does not have to be large enough to contain your programs.

Shared memory pool
A shared memeoy is a pool in which multiple subsystems can run jobs. Using shared memory pools allows the system to distribute similar jobs across multiple subsystems, still allowing these jobs to run in the same memory pool. You can specify 63 of the 64 shared memory pools that are defined on the system for use when creating subsystem descriptions. The machine pool is reserved for system use. Shared pools are either special or general; the machine pool and base pool are considered special shared pools, and all other shared pools are considered general shared pools.

Output queues

Output queues are areas where printer output files (also called spooled files) wait to be processed and sent to the printer. Printer output is created either by the system or by the user using a print file. A **print file** is similar to a template or a guideline where the default values for the attributes of printer output are set. It is the beginning of the printer output life cycle.

The print file contains the output queue (OUTQ) and print device (DEV)attributes, which dictate how the printer output is to be directed. The default settings are usually *JOB, meaning that the job attributes of the output queue and printer device determine how the printer output is directed. The job attributes of the output queue and printer device settings are based on information obtained when the job is created. This is based on information from the user profile the job is running under, the job description, the workstation device description, and the default printer(QPRTDEV).

When the printer output is ready to be created, the system checks the print file and the job attributes (in this order) to see what output queue will process the printer output and which printer device the system will use. You can change the parameters of the output queue (OUTQ) and printer device (DEV) at the time the job is submitted or at job run-time to bypass extended processing. For example, the user can set the print file output queue to a specific queue and set the printer device to their specific printer in the print file at job initiation for the changes to take effect immediately. In doing this, the printer output does not have to go through the job attributes to find the output queue and printer device it will use. If a specified output queue cannot be found, the printer output will be directed to QGPL/QPRINT. For more information on how printer output is created, see Chapter 1 of the Printer Device Programming manual.

Printer output files (also called spooled files) are files that hold information waiting to be printed or processed. The printer output file (also called spooled files) holds important attributes that define the position of the printer output on the queue with relation to other printer output. The position is defined by the priority, status, and schedule attributes.

Output queue
An **output queue** is an object that contains a list of printer output files (also called spooled files) to be written to an output device. The output queue carries important attributes that determine the order in which printer output is processed and the authority needed to make changes to the printer output file.

Priority
Printer output that is waiting to process is moved to the output queue based on its priority (ranges from 1-9 where 1 is the highest priority).

Status
The current status of printer output. You can view this status from the General page in Output properties.

Schedule
The schedule attribute tells when the file should start physical printing of the output data.

Immediate
Print immediately, even if the printer output file (also called spooled files) is not closed.
File end (default)
Printing begins as soon as the printer output file (also called spooled files) is closed.
Job end
Printing begins when the job ends.

Once the printer output file (also called spooled files) is ready to be printed, a writer job, a job that processes the printer output from the output queue to the printer device, takes data from the printer output file (also called spooled files) and sends it to the designated printer.

Attributes of an output queue

The output queue controls how printer output files (also called spooled files) are processed and who has the authority to perform actions on the output queue and associated printer output.

The order of files attribute determines how the printer output will leave the output queue to be processed. The two ways to configure the output queue, either by the job number or by the first in, first out (FIFO) rule.

Because most of the information that you print on the iSeries system is created as printer output, security is necessary to prevent unauthorized users access to confidential or sensitive material. Authority to check, data authorization, operator control, spool control, or being the owner allows you to access and makes changes on an output queue or printer output file (also called spooled files). You need one of the following authorities to perform any action on an output queue or printer output:

Authority to check. You must be the owner of the queue or have data authorization.

Display data. When this authority is set to *YES, it allows you to perform such actions as viewing, moving, sending output to another system, and copying printer output.

Operator control. If this attribute is set to *YES, users with *JOBCTL special authority are authorized to perform actions like hold, release, and delete printer output from the output queue. Other actions on printer output, output queues, and writers are allowed as well and are documented in the Security Reference Manual.

Spool control. Allows the user to perform all operations on printer output. The user must have *EXECUTE authority to the library the output queue is located in to perform any actions on the output queue.

Owner. This allows the user who owns the output queue to change or delete printer output.

Note: The default authority to the output queue is *USE public authority. Display Data authority is set to *NO (meaning not just anyone can view printer output). Authority to check is *OWNER (so the output queue owner can manipulate the printer output). Operator Control is set to *YES (meaning a user with *JOBCTL can hold, release, and delete printer output).

For more information on authorities needed to work with output queues, see Appendix D in the Security Reference Manual

.

Order of Files: The **order of files** attribute determines the sequence in which the printer output files (also called spooled files) are placed and processed on the output queue. Two ways to configure the output queue are by **job number** and **first in, first out (FIFO)**.

Job number
The queue entries for the printer output file (also called spooled file) are sorted in priority sequence using the job number of the job that created the printer output file.

First in, first out
New printer output files (also called spooled files)that enter the queue are placed after all other printer output files that have the same priority.

Note: You can only change the output queue order of files attribute when no printer output files are on the queue.

Status of printer output

The status of a printer output file (also called spooled files) determines where you will see it in the output queue. The following statuses are listed from the bottom of the output queue to the top.

Still being created

The printer output file is being created.

Printed and kept

The data in the printer output file has been printed, but has been saved to be used later.

Held

The printer output file is held, preventing it from being processed by a writer job.

Not scheduled to print yet

The creation of the printer output file is complete, but it is not eligible to be printed. This is only seen when the schedule attribute of the printer output file is set to *JOBEND. This means the job that owns the printer output file must end before the printer output file is allowed to be processed by a writer job.

Page limit exceeded

The file exceeds the maximum number of pages allowed to be printed by a writer job. This status is only seen if the output queue is active to a writer job.

Ready

The printer output file is waiting to be processed by the writer job.

The following statuses are seen when the output queue is active to a writer job (being processed by a writer job) and will be seen at the top of the output queue.

Converting for printer

The printer output file is in the process of being transformed (made ready) for the printer device.

»

Printing

The contents of the printer output file are being sent to the printer device.

Sent to printer

The contents of the printer output file are being printed. The operating system is waiting for confirmation that the printer output file is done printing.

Being sent

The printer output file is being transferred from one system to another system.

Message waiting

The writer job has encountered a problem, such as out of paper or a paper jam, where it may not be able to proceed printing. When this condition occurs, sometimes operator intervention will be required.

Finished printing

The printer output file has been deleted. Note, the printer output file may or may not have been printed.

How work gets done

Use this information to learn about what work is, what needs to be set up before work can begin, how work travels through the system, and what happens to work once it is done running.

- What work is
- What happens before work enters the system
- How work enters the system
- How work gets processed
- How work leaves the system

For more detailed information on the concepts of Work Management, see The structure of your system.

What work is

On the iSeries server, work is always being done, whether you initiate it or the system initiates it. Work is done when you power on your system, when you open a file, or when you query a database. Any action done on the iSeries server has some type of work being performed to complete it.

Each piece of work on the system is performed by a job. A job can be as simple as an application that waits for a user to call it or it can be as complex as a system query to monitor the number of users on the system every hour that runs constantly. Some jobs, specifically batch and interactive jobs, have job descriptions associated with them that tell when and where the job will run.

Jobs are made up of programs that perform certain functions. There is no limit to the amount of functions a job performs. A job contains the step-by-step instructions that must be completed for work to be done. The programs that make up the job run in a specific order. For example, program A needs to run before program B can begin.

»

Threads

«

help a job complete its work. An active job contains at least one thread. When a job contains multiple threads, it has the ability to do more than one thing at once. For example, one thread can go out and do calculations while another thread waits for more data to process.

For more detailed information on jobs and job types on the iSeries server, see Jobs.

What happens before work enters the system

All jobs, with the exception of system jobs, run within subsystems. For work to start in an active subsystem, memory pools and at least one source of work entry point need to be established. Job queues are an example of a source of work. The iSeries server ships with a default set of job queues, subsystems, and memory pools, which can allow work to begin as soon as the system is powered on.

You can tailor the subsystem and memory pool configurations to optimize your iSeries servers capabilities and performance. For example, if batch jobs are critical to the success of your business, you may want to allocate more memory for them to run. Or, you may determine that the number of jobs running at one time in your Qbatch subsystem should be lower so that those jobs can use the maximum amount of resources to run. Also, you can create job queues, subsystems, and memory pools specifically designed to complete specific types of work. For example, you can create a job queue called Nightreps, where nightly batch reports are sent to a subsystem called Nightrep that allocates memory exclusively for running these batch jobs.

To learn more about job queues, subsystems, and memory pools, see the The structure of your system . For more information on what IBM supports for work management, see **Appendix C. IBM-Supplied Object Contents** in the Work Management

manual and What's New for V5R2.

How work enters the system

Work entries identify the sources where jobs enter a subsystem to become available to run. Each type of job on iSeries has different types of work entries it uses.

Most batch jobs use job queues to enter the subsystem. Job queue entries are the mechanism through which a job queue is defined as a source of work to a subsystem.

Work entries are kept in the subsystem description. If a subsystem description does not have a work entry for the type of work being done, the job cannot run in that subsystem. The IBM-shipped subsystems have default work entries in the subsystem descriptions. Keep in mind, some of the default work entries that ship with the subsystems are already allocated to run specific jobs. For example, in the QCMN subsystem one of the communications work entries is set up to run the iSeries Access server.

For more information on how work enters the system, see work entries in Chapter 4 of the Work Management

manual.

How work gets processed

When the iSeries server is started, a subsystem monitor job begins running. The subsystem monitor job controls the jobs within subsystems. It also starts and ends work, as well as manages the resources for work in the subsystem. Work (or jobs) enters a subsystem through work entries where it becomes active and eligible to run. Work can only be completed when the subsystem is allocated memory to run. Memory is allocated to the subsystem by a memory pool.

How the subsystem description helps process work

Like a job, a subsystem has a description, called a subsystem description. The subsystem description contains important information that tells how, where, how much work can be active in a subsystem at one time, and which resources it can use to perform the work.

Routing entry
A **routing entry** exists within the subsystem description that tells the subsystem what memory pool to run the job in, what program to run for the job, and which class object to use to run the job. For more information about routing entries, see chapter 4 in the Work Management

manual.

Class Object
The **Class** object defines the run priority, default wait time, timeslice, and other attributes. The **run priority** is important because it determines when a job will get processor time in order to run. The run priority scale goes from 0 to 99, with 0 being the highest priority. (Only system jobs are given priority of 0 because they are the jobs that run the iSeries server.)

When a job enters the subsystem, the subsystem tries to match the **routing data** with the compare value in the routing entry. If the routing data and the compare value in a routing entry match, the routing entry is assigned to the job. If a match is not made, the job ends.

Another factor that affects when a job runs in the subsystem is the number of jobs allowed to be active in the subsystem at one time (also known as **maximum active jobs** in the subsystem). When the maximum number of active jobs in a subsystem has been met, no more jobs can enter the subsystem until existing active jobs complete running. Memory has to be allocated to the subsystem for a job to run. **Memory pool activity levels** tell the iSeries server how many threads can be active within a memory pool. Remember, an active job contains at least one thread. When the memory pool activity level has been reached, the job has to wait for another thread to give up its use of the activity level. A job can be active in a subsystem and not be running.

Note: Do not confuse the subsystem maximum active jobs with the memory pool activity level.

For more information on jobs, subsystems, and memory pools, see the Work Management

manual.

How work leaves the system

The output queue works similarly to a job queue in that it schedules output to be printed. Both the printer output and the output queue carry attributes that are used to print the information.

Printer output holds output data waiting to be processed, such as information waiting to be printed. Printer output also holds important information used to schedule when it will be printed. Printer output attributes include the output queue in which the printer output will reside, the priority, the status and the schedule of the printer output.

The output queue contains attributes of its own that determine the order in which the printer output files are processed. It also contains the authority needed to make changes to the printer output and the output queue.

When the printer output is ready to be sent to the printer it is picked up by a writer job. The writer job takes the data from printer output and prepares it to be printed.

For details about how the output queue gets selected see Controlling print activity in Chapter 1 of the Printer Device Programming

manual.

You can create specific output queues or use the output queues shipped with the system. For more detailed information, see Creating an output queue.

Troubleshoot Work Management

When a job does not appear to be processing efficiently on your iSeries server, it could be that the job is hung, or that it is just performing poorly. In each case, there are some diagnosis and recovery actions that can assist you in troubleshooting the problem. See the following topics for details.

- **My job is hung**
- **My job is experiencing poor performance**

My job is hung

The following are possible reasons why a job might be hung:

Job is waiting to get a lock on an object

How to diagnose:	View the status of the job in iSeries Navigator; see Determining the status of a job. A job that is waiting to get a lock will have a status of *Waiting for lock*.
Recovery:	View the list of locked objects for the job to determine which object the job is waiting to get a lock on; see Details: Active job actions. Then use the Lock Holders action against the object to determine which job already holds the lock. You then need to determine why this job is holding the lock, and what can be done to release the lock. In V5R2, status values can have additional information on the properties pages. For example, the status waiting for a lock on the Properties page shows you what object is associated with the lock request.

Job is held

How to diagnose:	View the status of the job in iSeries Navigator; see Determining the status of a job
Recovery:	Right click on the job and select *Release*.

The following are possible reasons why a job on a job queue might be hung:

Job queue is held

How to diagnose:	View the status of the job queue in iSeries Navigator;
Recovery:	1. Move the job to a job queue that is not held, see Moving jobs to different job queues. 2. Release the job queue. To do so, right-click the job and select *Release*.

Job queue has not been allocated by an active subsystem

How to diagnose: View the status of the job queue in iSeries Navigator.

Recovery:

1. Move the job to a job queue that is allocated by an active subsystem, see Moving jobs to different job queues.
2. Start a subsystem which contains a job queue entry for this job queue, see How subsystems start.
3. Add a job queue entry for this job queue to an active subsystem using the Add Job Queue Entry (ADDJOBQE) command.

Subsystem maximum has been reached

How to diagnose: View the maximum active jobs value for the subsystem in iSeries Navigator. To do so, right-click on the subsystem and select *Properties*

Recovery:

1. Move the job to a different job queue, see Moving jobs to different job queues.
2. Increase the maximum value. To do so, use the Change Subsystem Description (CHGSBSD) command.

Job queue maximum has been reached

How to diagnose: View the maximum active jobs value for the job queue in iSeries Navigator. To do so, right-click on the job queue and select *Properties*. Then select the **Activity** tab.

Recovery:

1. Move the job to a different job queue; see Moving jobs to different job queues.
2. Increase the maximum value. To do so, use the Change Job Queue Entry (CHGJOBQE) command.

Maximum value for the priority level has been reached

How to diagnose: Determine the job queue priority of the job by viewing its properties. Then view the maximum active jobs by job priority values for the job queue in iSeries Navigator. To do so, right-click the job queue and select *Properties*. Then select the Activity tab and click the Advanced button.

Recovery:

1. Move the job to a different job queue; see Moving jobs to different job queues.
2. Change the job queue priority of the job; see Changing the priority of a job within a job queue.
3. Increase the maximum value. To do so, use the Change Job Queue Entry (CHGJOBQE) command.

My job is experiencing poor performance

The following are possible reasons why a job might experience poor performance:

Insufficient memory

How to diagnose:
: View the properties of the job to determine which memory pool the job is running in. Then view the properties of the memory pool in iSeries Navigator, see Checking memory pool usage. A high rate of faulting in a pool indicates that there is not enough memory in the pool, or that too many jobs are in the pool competing for the memory.

Recovery:
1. Turn on the system tuner if you are not already using it. The system value QPFRADJ automatically adjusts memory pools and activity levels.
2. If possible, manually tune the pool you are working with by increasing the amount of memory in the pool or reducing the activity level for the memory pool. You may also want to check the machine pool to verify that the amount of memory being used is not affecting all jobs on the system.

Activity level too low

How to diagnose:
: View the properties of the job to determine its status and which memory pool the job is running in. If the job shows a status of *Waiting for activity level*, then view the properties of the memory pool in iSeries Navigator, see Checking memory pool usage. A high rate of transitions to the ineligible state in a pool indicates that too many jobs in the pool are competing for the memory.

Recovery:
1. Turn on the system tuner if you are not already using it. The system value QPFRADJ automatically adjusts memory pools and activity levels.
2. Manually tune the pool by increasing the activity level for the memory pool.

Insufficient CPU resource

How to diagnose:
: View the CPU % column for the job and other jobs in the Active Jobs list of iSeries Navigator. If the system is very busy, your job may not be getting enough CPU resource to complete its work.

Recovery:
1. If possible, end or hold unnecessary work on the system.
2. If a few jobs are CPU intensive, change the run priority of these jobs (a higher the run priority value equals a lower run priority for the job).

Memory pool paging option

How to diagnose:
: If an application is disk intensive, if the CPU is under utilized and if there is sufficient memory, the use of expert cache may be beneficial.

Recovery:
: The expert cache can be turned on in iSeries Navigator by changing the Paging option for a shared memory pool to Calculated. The Paging option is located on the **Configuration** tab of the memory pool's **Properties** page and is only available on shared pools(not private pools).

Low job run priority

How to diagnose:
: View the job's properties to determine the run priority of a job relative to other jobs on the system.

Recovery:
: If the job has a low run priority (higher number) relative to other jobs and is not using much CPU because the higher priority (lower number) jobs are using most of the CPU resource, you might need to increase the job's run priority, see Job properties. Also, on a system with high CPU utilization and a job with a low run priority, setting the Dynamically adjust job priorities within priority bands (QDYNPTYSCD) and the Dynamically adjust job priorities of interactive jobs (QDYNPTYADJ) system values may be useful.

For more information on performance, see Performance. If you want more information on how to tune performance on your system, see Tune performance.

Printed in U.S.A.