

IBM i  
7.3

*Systems management*  
*Work management*

**IBM**

**Note**

Before using this information and the product it supports, read the information in [“Notices” on page 197.](#)

This document may contain references to Licensed Internal Code. Licensed Internal Code is Machine Code and is licensed to you under the terms of the IBM License Agreement for Machine Code.

© **Copyright International Business Machines Corporation 2004, 2015.**

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

---

# Contents

- Work management..... 1**
- PDF file for Work management..... 1
- What's new for IBM i 7.3..... 2
- Introduction to work management..... 2
  - Your system as a business..... 2
  - A job's life..... 3
    - Submitting a job..... 4
    - The job enters the job queue..... 4
    - The job enters the subsystem..... 4
    - The subsystem uses memory from the memory pool to run the job..... 5
    - The job finishes and moves to the output queue..... 5
  - How work gets done..... 5
    - What work is..... 5
    - What happens before work enters the system..... 6
    - How work enters the system..... 6
    - How work gets processed..... 6
    - How work leaves the system..... 7
- Concepts..... 7
  - The structure of your system..... 7
    - Subsystems shipped with the system..... 7
    - Start-up programs..... 8
      - What happens during the IPL..... 9
    - Types of start-ups..... 9
    - Powering down your system..... 9
    - IBM Navigator for i..... 10
  - Subsystems ..... 10
    - The controlling subsystem..... 10
    - Why consider multiple subsystems..... 11
    - Subsystem description..... 11
      - Subsystem description attributes..... 12
      - Work entries..... 12
      - Routing entries..... 15
    - How a subsystem starts..... 18
      - How workstation devices are allocated..... 19
      - Scenario: Workstation allocation..... 20
- Memory pools..... 21
  - Types of memory pools..... 22
  - Pool numbering schemes..... 23
  - Memory pool allocation..... 25
  - Memory pool activity level..... 26
- Jobs..... 27
  - Proper authority..... 27
  - Job characteristics..... 27
    - Job name syntax..... 28
    - Job Attributes..... 28
    - Job description..... 28
    - Job descriptions and security..... 29
    - Call stacks..... 30
    - Class object..... 30
    - Job user identity..... 32
    - Job user identity examples..... 32

Threads.....	33
Locked objects.....	35
Job types.....	36
Autostart jobs.....	36
Batch jobs.....	37
Communication jobs.....	38
Interactive jobs.....	39
Prestart jobs.....	44
Reader and writer jobs.....	48
Server jobs.....	48
System jobs.....	49
Job scheduling options.....	53
Management Central scheduler.....	53
Job schedule entries.....	53
Examples: job schedule entry.....	54
The submit job command.....	55
Job scheduler considerations.....	56
Job scheduling and system availability.....	57
Job queues.....	57
Ordered list.....	58
How a job queue works.....	59
How jobs are taken from a job queue.....	59
Job queue entry.....	60
How job queues are allocated to a subsystem.....	61
Multiple job queues.....	61
How jobs are taken from multiple job queues.....	62
Job queue security.....	62
Output queues.....	63
Attributes of an output queue.....	64
Order of files.....	64
Spooled files.....	65
Output spooling.....	65
Output queues and spooled files.....	66
Default system output queues.....	66
Spooling writers.....	67
Spooling writer commands.....	67
Input spooling.....	67
Job input commands.....	69
Inline data files.....	70
Considerations for opening inline data files.....	71
Job logs.....	71
How job logs are created.....	72
Job log pending.....	73
Job log server.....	74
Job log display characteristics.....	75
Job log headings.....	75
Messages.....	75
Interactive job logs.....	76
QHST History Log.....	77
Format of the History Log.....	78
Performance information and QHST.....	79
Spooled files.....	79
Job accounting.....	79
How job accounting works.....	81
Job Accounting operating characteristics.....	83
Accounting Journal Processing.....	83
When to use job accounting.....	84
Security and job accounting.....	84

About the accounting code.....	85
Resource accounting.....	86
Resource accounting data.....	86
Prestart communications jobs and job accounting.....	87
System job processing for job accounting.....	88
Batch processing and job accounting.....	88
Interactive processing and job accounting.....	89
Printer file accounting.....	89
Journal entries for job accounting.....	89
Job accounting journal entry field information.....	90
Printer file accounting data for direct print and spooled print.....	93
Managing work.....	96
Calling a special IPL recovery program.....	96
Monitoring system activity.....	96
Checking memory pool usage.....	97
Controlling levels of system activity.....	98
Examples: activity control relationships.....	100
Determining the status of a job.....	101
Monitoring a subsystem.....	101
IBM Navigator for i.....	101
Determining the number of subsystems using a memory pool.....	101
IBM Navigator for i.....	101
Character-based interface.....	102
Viewing job performance statistics .....	102
Viewing overall system status.....	102
Checking disk status.....	103
Managing jobs.....	103
Common job tasks.....	103
Starting a job.....	103
Ending a job.....	104
Finding jobs .....	106
Viewing jobs on the job queue.....	107
Viewing jobs in the subsystem.....	107
Viewing job attributes.....	107
Viewing call stacks.....	108
Placing a job on the job queue.....	109
Moving a job to a different job queue.....	109
Moving a job up in priority.....	110
Tips for setting job priorities.....	111
Submitting a job once.....	111
Viewing job affinity information.....	111
Managing job descriptions.....	112
Creating a job description.....	112
Changing a job description.....	113
Using a job description.....	113
Controlling the job attribute source.....	113
Deleting a job description.....	114
Manage batch jobs.....	114
Submitting a batch job.....	114
Starting a batch job that is waiting in the job queue.....	116
Managing interactive jobs.....	116
Controlling inactive jobs and workstations.....	116
Ending interactive jobs.....	117
Disconnecting all jobs from a device.....	118
Job disconnection considerations.....	118
Avoiding a long-running function from a workstation.....	119
Managing prestart jobs.....	119
Starting a prestart job .....	119

Queueing or rejecting program start requests.....	120
Tuning prestart job entries.....	120
Changing job attributes for prestart jobs.....	123
Ending a prestart job.....	125
Managing job class objects.....	125
Creating a class object.....	125
Changing a class object.....	126
Managing threads.....	126
Viewing threads running under a specific job.....	126
What you can do with threads.....	127
Viewing thread properties.....	127
Ending or deleting threads.....	128
Managing job scheduling.....	129
Scheduling a batch job using IBM Navigator for i.....	129
Scheduling a job using Management Central Scheduler.....	129
Working with job schedule entries.....	130
Adding a job schedule entry.....	130
Changing a job schedule entry.....	130
Holding a job schedule entry.....	131
Printing a list of job scheduled entries.....	131
Releasing a job schedule entry.....	131
Removing a job schedule entry.....	131
Managing subsystems.....	132
Common subsystem tasks.....	132
Viewing subsystem attributes.....	132
Stopping a subsystem.....	133
Starting a subsystem.....	134
Creating a subsystem description.....	134
Adding autostart job entries.....	135
Adding communications entries.....	136
Adding job queue entries.....	136
Adding prestart job entries.....	137
Adding routing entries.....	137
Adding workstation entries.....	137
Creating a sign-on display file.....	138
Specifying the new sign-on display .....	139
Changing a subsystem description.....	139
Changing autostart job entries.....	139
Changing communication entries.....	140
Changing job queue entries.....	140
Changing prestart entries.....	141
Changing routing entries.....	141
Changing workstation entries.....	141
Changing the sign-on display.....	142
Deleting a subsystem description.....	142
Removing autostart job entries.....	143
Removing communication entries.....	143
Removing job queue entries.....	143
Removing prestart job entries.....	144
Removing routing entries.....	144
Removing workstation entries.....	144
Configuring an interactive subsystem.....	145
Creating a library.....	145
Creating a class.....	145
Creating the subsystem description.....	145
Creating a job queue.....	146
Adding a routing entry.....	146
Adding workstation entries.....	146

Customizing QINTER.....	146
Configuring the console.....	147
Assigning users to a specific subsystem.....	147
Configuring a server subsystem.....	148
Creating a user-defined server subsystem.....	149
Routing server jobs based on client IP address.....	150
Routing server jobs by user.....	151
Creating a controlling subsystem.....	152
Placing the system in restricted state.....	153
Managing memory pools.....	154
Viewing memory pool information.....	154
IBM Navigator for i.....	154
Character-based interface.....	154
Determining the number of subsystems using a memory pool.....	155
IBM Navigator for i.....	155
Character-based interface.....	155
Determining the number of jobs in a memory pool.....	155
Determining in which pool a single job is running.....	156
IBM Navigator for i.....	156
Managing tuning parameters for shared pools.....	157
IBM Navigator for i.....	157
Character-based interface.....	157
Managing a pool's configuration.....	157
IBM Navigator for i.....	158
Character-based interface.....	158
Changing memory pool size.....	158
IBM Navigator for i.....	158
Character-based interface.....	159
Change the size of a shared pool.....	159
Creating a private memory pool.....	159
Managing job queues.....	160
Assigning the job queue to the subsystem.....	160
How a subsystem handles several job queues.....	160
Changing the number of jobs running simultaneously in a job queue.....	161
Clearing a job queue.....	161
IBM Navigator for i.....	162
Character-based interface.....	162
Creating job queues.....	162
Deleting a job queue.....	162
Determining which subsystem has a job queue allocated.....	163
IBM Navigator for i.....	163
Character-based interface.....	163
Holding a job queue.....	163
IBM Navigator for i.....	164
Releasing a job queue.....	164
IBM Navigator for i.....	164
Character-based interface.....	164
Moving a job to a different job queue.....	164
IBM Navigator for i.....	165
Placing a job on the job queue.....	165
IBM Navigator for i.....	165
Character-based interface.....	165
Searching all job queues for a specific job.....	166
IBM Navigator for i.....	166
Character-based interface.....	166
Find a job when you do not know the name of the job queue.....	166
Specifying the priority for the job queue.....	166
Managing output queues.....	167

Creating an output queue.....	167
Assigning the output queue to a job or job queue.....	167
IBM Navigator for i.....	167
Character-based interface.....	167
Accessing printer output.....	168
IBM Navigator for i.....	168
Clearing output queues.....	168
IBM Navigator for i.....	168
Character-based interface.....	168
Deleting an output queue.....	168
Viewing output queues on the system.....	169
Managing job logs.....	169
Managing the job log server.....	169
Reconfiguring the job log server.....	169
Ending the job log server.....	170
Starting the job log server.....	171
IBM Navigator for i.....	171
Character-based interface.....	171
How to display job logs.....	171
IBM Navigator for i.....	172
What to do when the job log does not display.....	172
Specifying the output queue for a job log.....	173
Stopping production of a specific job log.....	173
Preventing the production of a job log.....	174
Controlling information in a job log.....	175
Changing the log level of a job.....	175
Controlling batch job log information.....	176
Deleting job log output files.....	177
Producing printer output from job log pending.....	178
Cleaning up job log pending.....	178
IBM Navigator for i.....	178
Character-based interface.....	179
Managing job accounting.....	179
Setting up job accounting.....	179
Controlling the assignment of accounting codes.....	180
Displaying the data collected.....	180
Converting job accounting journal entries.....	181
Recovering and job accounting.....	182
Damaged job accounting journal or journal receiver.....	183
Accessing the CPF1303 Message.....	183
Managing workload groups.....	184
Setting up workload groups.....	184
Displaying workload groups.....	185
Auditing workload groups.....	185
Reference.....	186
Group jobs.....	186
Attention key handling program.....	189
Group job performance tips.....	191
Troubleshooting for work management.....	192
My job is hung.....	192
My job is experiencing poor performance.....	193
Prestart job investigation.....	195
Related information for Work management.....	196
<b>Notices.....</b>	<b>197</b>
Programming interface information.....	198
Trademarks.....	198



Terms and conditions..... 199



---

# Work management

Work management is an important building block within the IBM® i operating system.

Its functions are the foundation through which all work enters the system, is processed, run, and completed. 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 IBM i product is set up and ready to use. Most users do not need to change the default settings. However, if you need to tailor the work management piece to fit your company, you 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 system.

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

**Note:** In addition, you can work with work management using IBM Navigator for i. This allows you to work with work management functions using a Web browser. For more information, see [IBM Navigator for i tasks on the Web](#).

---

## PDF file for Work management

You can view and print a PDF file of this information.

To view or download the PDF version of this document, select [Work management](#).

You can view or download these related topics:


- [Performance](#) contains the following topics:
  - Planning for performance
  - Managing system performance
  - Applications for performance management

### Saving PDF files

To save a PDF on your workstation for viewing or printing:

1. Right-click the PDF link in your browser.
2. Click the option that saves the PDF locally.
3. Navigate to the directory in which you want to save the PDF.
4. Click **Save**.

### Downloading Adobe Reader

You need Adobe Reader installed on your system to view or print these PDFs. You can download a free copy from the [Adobe Web site](http://www.adobe.com/products/acrobat/readstep.html) ([www.adobe.com/products/acrobat/readstep.html](http://www.adobe.com/products/acrobat/readstep.html)) .



## What's new for IBM i 7.3

---

### How to see what's new or changed

The [Configuring a server subsystem](#) section was added to the Managing subsystems topic. This new section has information on setting up a user-defined server subsystem and configuring servers to route their work to different subsystems.

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.

In PDF files, you might see revision bars (|) in the left margin of new and changed information.

To find other information about what's new or changed this release, see the [Memo to users](#).

## Introduction to work management

---

Work management supports the commands and internal functions necessary to control system operation and the daily workload on the system. In addition, work management contains the functions that you need to distribute resources for your applications so that your system can handle your applications.

The purpose of your system is to perform work. Work enters, work is processed, and work leaves the system. If you think of work management in these three terms, work management will be easier to understand. Work management describes where work enters the system, where and with what resources work is processed, and where output from work goes.

Are you new to work management? The topic collection under the subject Introduction to work management is designed to provide you with several different overall perspectives of work management. In this way, you should be able to get a solid foundation in the underlying principles of work management, regardless of your systems background.

### Your system as a business

To make grasping the overall concept of work management easier, try comparing your system with a business.

A simple system can be compared to a small business, and a complex system can be compared to a shopping mall. Assume there is a small store in the business of building hand-crafted wood furniture. *Work enters*, such as orders for small tables, chairs, and bookshelves. *Work is processed*, the carpenter calls the customers to confirm the order, and they are consulted on design points including style, size, and color. The carpenter designs each piece of furniture, gathers the necessary materials, and then builds the furniture. After the furniture is completed, it is delivered: *work leaves*.

Since a complex system is a combination of many simple systems, a comparable example of a complex system is a shopping mall, many small and large businesses in one area. Maybe the carpenter has a business in the Northwest corner of the mall and a baker has a business along the East strip. The baker and the carpenter have different input and different output, that is, their orders and their products are very different. In addition, the time it takes each business to process their work is quite different, and their users know and understand that.

### Work management terms

A complex system (shopping mall) is a compilation of many simple systems (stores). These simple systems are called *subsystems*.

Any piece of work within the business is considered a *job*. An example of a piece of work might be a customer letter, a telephone call, an order, or nightly cleanup. The same can be said about the IBM i product. On the system, each job has a unique name.

A *job description* describes how to handle the work coming into the subsystem. Job descriptions contain pieces of information such as user IDs, job queues, and routing data. Information in the job description might compare to descriptions of jobs in a small business.

**What does the business look like?** Every store has blueprints or store plans. These plans are really just descriptions, in varying detail, of the physical makeup of the business. Maybe the business has a store with: 2 floors, 5 doors, 3 mailboxes, and 2 telephones. On your system, a *subsystem description* contains all the information about the subsystem.

**Where does the work come from?** For the carpenter, the work comes from customer calls, from references, and from people that stop in. On your system, the work can come from many places. Examples include job queues, workstations, communications, autostart jobs, and prestart jobs.

**Where do they find the space?** Within the mall, each business (subsystem) has a certain amount of floor space. On the system, *memory pools* allow you to control the main storage (or floor space) each subsystem (business) gets to do its work. The more floor space a store (subsystem) has, the more customers, or jobs, can fit in the store.

**How does the work come in?** Customers that cannot find the store they need may find an information booth to help send them in the right direction. The same is true on your system. *Routing entries* are similar to store directories or an information booth. After the routing entry is found, it guides the job to its correct place. The routing entry needs to be found first, however. That is done through *routing data*. Routing data is what the job uses to find the right routing entry.

**How is the work treated?** A carpenter needs to place a priority on each job. The chair due at the end of the week should be done before the bookshelf due at the end of the month. On your system, *classes* provide information about how the job is handled while in the subsystem. This information includes priority while running, maximum storage, maximum CPU time, and time slice. Each of these attributes contribute to how and when a job is processed.

Just as there are rules that affect all the stores in the mall, there are rules that affect all the subsystems on your system. An example of these rules is a *system value*. System values are pieces of information that apply to the whole system. System values include information such as, date and time, configuration information, sign-on information, system security and, storage handling.

Customers in a mall each have information specific to them. On your system, the *user profile* holds information specific to a particular user. Similar to a customer's credit card, a user profile gives that user specific authorities and assigns the user attributes for that user's jobs. These *job attributes* provide information that includes, but is not limited to, the job description, the output queue or printer device, the message queue, the accounting code, and the scheduling priority.

## A job's life

To understand the basics of IBM i work management, follow a simple batch job as it moves through the system.

The life of a simple batch job begins when you submit it to the system. The job is then sent to a job queue where it waits to enter a subsystem where it can run. After the job moves to the subsystem it is allocated memory in which to run. The printer output file (also called spooled files) is then sent to the output queue to await further instruction on what to do (for example, printing). While not every job follows this exact path, you can better understand how other work is completed on the system by learning more about this typical job life cycle.

**Submit the job > Job enters the job queue > Job enters the subsystem > The memory pool allocates memory to the subsystem > The job finishes and moves to the output queue**

## Submitting a job

When a job is submitted, it is created and enters the system. At this time, the attributes are given to the job.

The job description holds *attributes* that the job will use to go through the work management life cycle. These attributes include the user profile the job will start to run under, the request data (which tells the job what it will do), and the initial user portion of the library list, and so on. The job description also holds information that tells the job which job queue to enter and the routing data. The routing data is later used by the subsystem to find the routing entry that contains information needed for the job to start running. The output queue is also defined within the job description. It tells where printer output (also called spooled files) from a job will go.

After the job receives its values (initialization, customization) for its job attributes, it moves to the job queue where it waits to enter the subsystem.

## The job enters the job queue

*Job queues* are work entry points for batch jobs to enter the system. They can be thought of as "waiting rooms" for a subsystem.

A number of factors affect when the job is pulled off the job queue into the subsystem, such as the job priority on the job queue, the sequence number of the job queue, and the maximum active jobs. When all of these factors work together, the job will be pulled off the job queue to start running in the subsystem.

When the job enters the job queue, it is available to a subsystem that has the job queue allocated to it. Because subsystems can have more than one job queue feeding into them (however, job queues cannot feed into more than one subsystem), a sequence number in the subsystem determines when the subsystem processes a job queue. The subsystem looks at the sequence number of the job queue before the job priority of the jobs in the job queue. The subsystem uses the priority on the job queue to determine when a job can enter relative to other jobs on the job queue. The job priority and the maximum active jobs determine when a job enters the subsystem.

## The job enters the subsystem

*Subsystems* are operating environments where the system manages the resources that jobs use and controls the jobs that run within them. After jobs are running in the subsystem, the subsystem job carries out user requests on a job such as holding, releasing, and ending a job. When the job enters the subsystem it becomes active.

Like jobs, subsystems have descriptions that carry important information needed to complete the work. In the subsystem description is the routing entry. The *routing entry* references the class object, which contains the attributes that control the run-time environment. However, before the job can get its routing entry, the routing data must make a match with a compare value in the routing entry. If this association is not made, the job is not run.

After the association between the routing data and the routing entry is made, the class object the job uses is determined. Some of the attributes that control the run-time environment include the run priority, the time slice, the maximum wait time, the maximum processing time, the maximum temporary storage, and the maximum number of threads.

The subsystem description defines the memory pools that are allocated to the subsystem. The subsystem description also contains the maximum active jobs, which is the maximum number of active jobs at one time in the subsystem.

Until a job gets its activity level and is assigned a memory pool, it cannot run. The subsystem description, like the job description, carries information, such as the memory pool to use, the routing entry, the maximum active jobs, and the number of active jobs currently in the subsystem.

## The subsystem uses memory from the memory pool to run the job

*Memory* is a resource from the memory pool that the subsystem uses to run the job. The amount of memory in a memory pool, as well as how many other jobs are competing for memory, affect how efficiently a job runs.

*Memory pools* provide jobs with memory in which to run. Many factors affect how the job runs in the memory pool, such as the size of the memory pool, the activity level in the memory pool, and paging and faulting. The activity level in memory pools directly relates to the number of threads that are allowed to run in the memory pool at one time. Remember, every job has at least one active thread, but some can have multiple threads. Threads give a job the ability to do more than one thing at a time. For example, one thread can go out and do calculations while another thread waits for more data to process.

*Paging* is the movement of data in and out of memory, both synchronously and asynchronously. Pages can be written out to storage or removed from memory without being written if they have not been changed. Faulting causes paging to occur on the server. Faulting occurs when a referenced page, or piece of data, is not in memory. This causes programs to stop because they must wait for the data to be paged in.

Subsystems use different memory pools to support different types of jobs that run within them.

## The job finishes and moves to the output queue

A job's printer output (also called spooled files) is sent to an output queue where it waits to be sent to a printer or file. The output queue is similar to the job queue in that it controls how the output is made available to the printer. The output queue allows the user to control what files are printed first.

*Output queues* are areas where printer output 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 typically \*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 that the job is running under, the job description, the workstation device description, and the Default printer (QPRTDEV) system value.

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. If a specified output queue cannot be found, the printer output will be directed to QGPL/QPRINT.

After the printer output file 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 and sends it to the designated printer.

## How work gets done

This topic explains what work is, what needs to be set up before work begins, how work travels through the system, and what happens to work after it is done running.

### What work is

On your IBM i product, work is always being done, whether you initiate it or the system initiates it. Any action done on the system has some type of work being performed to complete it.

Work is done when you turn on your system, when you open a file, or when you query a database. 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 constantly running system query that monitors the number of users on the system every hour. 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.

## **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 IBM i product includes 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 the capabilities and performance of your IBM i product. 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.

## **How work enters the system**

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

For example, 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.

## **How work gets processed**

When the system 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 has 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 and tells the subsystem what program to run for the job, what memory pool to run the job in, and which class object to use to run the job.

### **Class Object**

The Class object defines the run priority, default wait time, time slice, and other attributes. The run priority is important because it determines when a job gets 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 system.)

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 in any routing entry, the job ends.

Another factor that affects when a job runs in the subsystem is the number of jobs that are 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 system 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. Thus, 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.

## 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 that is waiting to be processed, such as information waiting to be printed. Printer output also holds important information that is used to schedule when it is printed. Printer output attributes include the output queue in which the printer output resides, 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 that is 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 the printer output and prepares it to be printed.

## Concepts

---

Whether you are new to work management or have been using work management tools for years, these work management concepts might be useful for you.

## The structure of your system

After receiving your IBM i product, you might want to know what subsystems are included with the system, whether you need to change any start-up programs, and what kind of user interface you will work with.

### 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/library (QCTLSBSD) system value. The default configuration consists of the following subsystem descriptions:

Subsystem	Description
<b>Qbase (controlling subsystem)</b>	Qbase supports interactive, batch, and communications jobs. It has an autostart job, which automatically starts the Qusrwrk, Qserver, and Qspl subsystems.
<b>Qserver</b>	This is the file server subsystem.
<b>Qspl</b>	This is the spool subsystem that supports reader and writer jobs.

<b>Qsyswrk</b>	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.
<b>Qusrwrk</b>	This is the user work subsystem. It contains jobs that are started by servers to do work on behalf of a user.

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

<b>Subsystem</b>	<b>Description</b>
<b>Qctl (controlling subsystem)</b>	Qctl has an autostart job, which automatically starts the Qinter, Qbatch, Qcmn, Qusrwrk, Qserver and Qspl subsystems.
<b>Qinter</b>	This is the subsystem that supports interactive jobs, except those at the console.
<b>Qbatch</b>	This is the subsystem that supports batch jobs.
<b>Qcmn</b>	This is the subsystem that supports communications jobs, excluding TCP/IP communications jobs. These communications jobs are necessary for various communications protocols that the IBM i system supports.
<b>Qserver</b>	This is the file server subsystem.
<b>Qspl</b>	This is the spool subsystem that supports reader and writer jobs.
<b>Qsyswrk</b>	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.
<b>Qusrwrk</b>	This is the user work subsystem. It contains jobs that are started by servers to do work on behalf of a user.

The Qbase configuration gives the ability to run all of 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, 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 ending the Qinter subsystem.

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

## Start-up programs

QSTRUPPGM is the start-up program. This is a system value which specifies the name of the program called from an autostart job when the controlling subsystem is started. This program performs setup functions, such as starting subsystems and printers. This system value can only be changed by the security officer or by someone with security officer authority. A change to this system value takes effect the next time an IPL is performed.

QSTRUPPGM can have these values:

- QSTRUP QSYS: The program specified is run as a result of a transfer of control to it from the autostart job in the controlling subsystem.
- \*NONE: The autostart job ends normally without calling a program.

### Related information

[System values that control IPL](#)

## ***What happens during the IPL***

The default startup program QSYS/QSTRUP does the following:

- Starts the QSPL subsystem for spooled work.
- Releases the QS36MRT and QS36EVOKE job queues if they were held (these are used by the System/36 environment).
- Starts Operational Assistant cleanup, if allowed.
- Starts all print writers unless user specified not to on the IPL Options display.
- Starts the QSERVER and QUSRWRK subsystems.
- If the controlling subsystem is QCTL, it starts the QINTER, QBATCH, and QCMN subsystems.

## **Types of start-ups**

During an initial program load (IPL), system programs load from the designated load source device in the system auxiliary storage. The system hardware is also checked. The IBM i control panel displays a series of system reference codes that indicate its current status and warn you of any problems. When the IPL is finished, the character-based interface presents the sign-on display, and users are able to sign on with IBM Navigator for i.

There are several options for starting your system. You can:

- Start the system without making configuration changes. This is referred to as an *unattended IPL*.
- Change your system configuration during an IPL. This is referred to as an *attended IPL*.

Attended IPL's display various additional screens depending upon the options that you select on the IPL options display. These can include displays that allow you to change system values and other system attributes during the IPL, reconstruct access paths, verify the status of physical file restrictions, configure and name new devices, and specify options for the operating environment.

- Change the type of IPL from your system control panel.
- Schedule a system shut down and restart.

General problems during an IPL is referred to as an *abnormal IPL*.

For more information about IPL and system shut down, see information about starting and stopping the system.

### **Related information**

[Starting and stopping the system](#)

## **Powering down your system**

You must be cautious when turning off your system. If you turn off the system without completing certain tasks, you can cause damage to data or cause the system to behave in unpredictable ways.

The following information center topics contain more information about safely powering down your system.

- How to safely shut down your system when integrated Windows servers are present
- Power down a system with logical partitions
- Power down System Exit Program
- Exit Program for Tailoring Power Off

### **Related information**

[Shutting down your IBM i system when integrated Windows servers are present](#)

[Power down a system with logical partitions](#)

[Power Down System Exit Program API](#)

[Exit Program for Tailoring Power Off API](#)

## IBM Navigator for i

IBM Navigator for i is a web based graphical interface. With IBM Navigator for i, you can manage and administer your systems from your browser. You can use IBM Navigator for i to accomplish most of the tasks associated with work management.

This interface has been designed to make you more productive. Therefore, it is recommended that you use IBM Navigator for i, which has online help to guide you. While this interface is being developed, you might still need to use a traditional emulator such as PC5250 to do some of your tasks. If a topic discusses such a task, you will be directed to use the character-based interface within the instructional steps of the topic.

### Related information

[Working with IBM Navigator for i](#)

[IBM Navigator for i](#)

## Subsystems

The subsystem is where work is processed on the system. A subsystem is a single, predefined operating environment through which the system coordinates the work flow and resource use. The system can contain several subsystems, all operating independently of each other. Subsystems manage resources.

All jobs, with the exception of system jobs, run within subsystems. 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 handles.

The run-time characteristics of a subsystem are defined in an object called a subsystem description. For example, if you want to permanently change the amount of work (number of jobs) coming from a job queue into a subsystem you only need to change the job queue entry in the subsystem description.

### Related tasks

[Common subsystem tasks](#)

This information discuss the most common tasks that you can perform on a subsystem.

[Creating a subsystem description](#)

You can create a subsystem description in two ways. You can copy an existing subsystem description and change it, or you can create an entirely new description.

### Related information

[Experience Report: Subsystem Configuration](#)

## The controlling subsystem

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 via the system console. It is identified in the Controlling subsystem/library (QCTLSBSD) system value.

IBM supplies two complete controlling subsystem descriptions: QBASE (the default controlling subsystem) and QCTL. Only one controlling subsystem can be active on the system at any time.

When the system is in the restricted condition, most of the activity on the system has ended, and only one workstation is active. The system must be in this condition for commands such as Save System (SAVSYS) or Reclaim Storage (RCLSTG) to run. Some programs for diagnosing equipment problems also require the system to be in a restricted condition. To end this condition, you must start the controlling subsystem again.

**Note:** There is also a batch restricted state in which one batch job can be active.

When all of the subsystems, including the controlling subsystem are ended, a restricted condition is created. You can end each subsystem individually or you can use the ENDSBS SBS(\*ALL) OPTION(\*IMMED).

**Important:** The system cannot reach the restricted state until there is only one job remaining in the controlling subsystem. Sometimes it may appear as though there is a single job remaining, but the system does not go into the restricted state. In this case you need to verify that there are no suspended system request jobs, suspended group jobs, or disconnected jobs on the remaining active display. Use the Work with Active Jobs (WRKACTJOB) command and press F14=Include to display any suspended or disconnected jobs. If these jobs exist, you need to end them in order for the system to reach the restricted state. The ENDSYS and ENDSBS functions send a CPI091C information message to the command issuer when this condition is detected.

### **Related tasks**

#### Creating a controlling subsystem

IBM supplies two complete controlling subsystem configurations: QBASE (the default controlling subsystem), and QCTL. Only one controlling subsystem can be active on the system at one time. Typically, the IBM supplied subsystem configurations should be sufficient for most business needs. However, you can create your own version of a controlling subsystem and configure it to more closely meet your company's unique needs.

#### Placing the system in restricted state

If all of the subsystems, including the controlling subsystem are ended, the system goes into a restricted condition. You can place the system in a restricted condition by using one of two commands from an interactive workstation.

### **Related information**

Experience Report: Restricted State

## **Why consider multiple subsystems**

As the number of users on the system increases, a single subsystem for a set of work is often insufficient. By dividing your users into multiple subsystems you gain several advantages.

### **Improved manageability of work**

You get better control over what work is running in each subsystem. For example, for server jobs, you might want to isolate all of the database server jobs to one subsystem, the remote command server jobs to a different subsystem, the DDM server jobs to yet a different subsystem and so on. Additionally, by using multiple subsystems you can isolate groups of jobs with their own memory pools. In this way, one group does not adversely impact other jobs.

### **Reduced downtime impact for users**

For example, if every Friday afternoon you must bring the system to the restricted state for backup purposes, you can gradually take users offline by ending one subsystem at a time.

### **Improved scalability and availability**

By having a single subsystem do work for fewer users, the subsystem is less busy and can be more responsive to the work requests it handles.

### **Improved error tolerance in interactive subsystems**

By spreading the work across multiple subsystems, should a network failure occur, multiple subsystems can manage the device recovery processing.

### **Improved interactive subsystem startup time**

You can keep the subsystem startup times shorter by subdividing the work across multiple subsystems.

### **Additional options for performance tuning**

By using multiple subsystems you can set up the subsystems with a small number of routing entries.

### **Related information**

Experience Report: Subsystem Configuration

## **Subsystem description**

A subsystem description is a system object that contains information defining the characteristics of an operating environment controlled by the system. The system-recognized identifier for the object type is

\*SBSD. A subsystem description defines how, where, and how much work enters a subsystem, and which resources the subsystem uses to perform the work. An active subsystem takes on the simple name of the subsystem description.

Like a set of detailed blueprints, each subsystem description is unique, containing the specific characteristics that describe the subsystem. The description includes where work can enter the subsystem, how much work the subsystem can handle, how much main storage (memory) is used, and how quickly jobs in the subsystem can run.

You can use a subsystem description supplied with your system (with or without making changes to it), or you can create your own.

### **Related tasks**

#### Changing a subsystem description

The Change Subsystem Description (CHGSBSD) command changes the operational attributes of the specified subsystem description. You can change the subsystem description while the subsystem is active. To change a subsystem description, use the character based interface.

#### Creating a subsystem description

You can create a subsystem description in two ways. You can copy an existing subsystem description and change it, or you can create an entirely new description.

### ***Subsystem description attributes***

Subsystem description attributes are common overall system attributes. When you create a subsystem, the first step is to define the subsystem attributes.

Subsystem attributes include:

- The name of the subsystem description and the library where it is stored
- All of the memory pool definitions that this subsystem uses

A subsystem definition can have a maximum of 10 memory pool definitions specified. Included in the subsystem definition are:

- Pool definition identifier: This is the identifier inside the subsystem description, of the storage pool definition.
- Size: This is the size of the storage pool expressed in KB (1K=1024 bytes) multiples and is the amount of main storage that the pool can use.
- Activity level: This is the maximum number of threads that can run at the same time in the pool.
- The maximum number of jobs that can be active in the subsystem at the same time
- A text description of the subsystem description
- The name and library of the sign-on display file that is used to show sign-on displays at work stations that are allocated to the subsystem
- A subsystem library name that you can use if you want to specify a library that should be entered ahead of other libraries in the system portion of the library list (This parameter allows you to use a secondary language library.)

Also included in the subsystem description is information about authority levels to the subsystem. This information is kept by Security and is not stored with the other attributes of the subsystem description. You can view the subsystem description authority by using the Display Object Authority (DSPOBJAUT) command.

### ***Work entries***

Work entries identify the sources where jobs can enter a subsystem. Specific types of work entries are used for different types of jobs. Work entries are part of the subsystem description.

The following information describes the different types of work entries and how to manage them. There are five types of work entries; autostart job entries, communication entries, job queue entries, prestart job entries, and workstation entries.

### *Autostart job entries*

Autostart job entries identify the autostart jobs to start as soon as the subsystem starts. When a subsystem starts, the system allocates several items and starts autostart and prestart jobs before it is ready for work.

The autostart jobs associated with a subsystem are automatically started each time the subsystem is started. An autostart job in the controlling subsystem can be used to start other subsystems (as does the IBM-supplied controlling subsystem). An autostart job is a batch job doing repetitive work.

For example: To call a special recovery program if the IPL senses that the previous system ending was abnormal, you can add an autostart job entry to the subsystem description for the controlling subsystem. This program checks the Previous system ending status (QABNORMSW) system value. For a normal system ending, the value of QABNORMSW is '0', and for an abnormal system ending, the value of QABNORMSW is '1'.

### **Related tasks**

#### Adding autostart job entries

You use the character-based interface to add an autostart job entry. An autostart job starts automatically when the associated subsystem 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.

#### Changing autostart job entries

You can specify a different job description for a previously defined autostart job entry. To change an autostart job entry, use the character-based interface

#### Removing autostart job entries

You can remove an autostart job entry from a subsystem description by using the character-based interface.

### *Communications entries*

The communications work entry identifies to the subsystem the sources for the communications job it processes. The job processing begins when the subsystem receives a communications program start request from a remote system and an appropriate routing entry is found for the request.

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 a communications batch job to run on system, a subsystem description containing a work entry for communications work must exist on the system.

### **Related tasks**

#### Adding communications entries

Each communication entry describes one or more communication device, device types, or remote location for which the subsystem starts jobs when program start requests are received. The subsystem can allocate a communication device if the device is not currently allocated to another subsystem or job. A communications device that is currently allocated may eventually be de-allocated, making it available to other subsystems. To add a communications entry to the subsystem description, use the character-based interface.

#### Changing communication entries

You can change the attributes of an existing communications entry in an existing subsystem description by using the character-based interface.

#### Removing communication entries



You can remove communication entries from the subsystem description by using the character-based interface. All jobs that are active through the communications entry being removed must be ended before this command can be run.

#### *Job queue entries*

Job queue entries in a subsystem description specify from which job queues a subsystem is to receive jobs. When the subsystem is started, the subsystem tries to allocate each job queue defined in the subsystem job queue entries.

For example, a job queue entry in the subsystem description QSYS/QBASE specifies that jobs can be started using the job queue QGPL/QBATCH. Jobs can be placed on a job queue even if the subsystem has not been started. When the subsystem QBASE is started, it processes the jobs on the queue. A subsystem description can specify the maximum number of jobs (batch or interactive) that can be processed at the same time. The number of jobs that can be active from any job queue is specified in the job queue entry.

### **Related tasks**

#### Adding job queue entries

A job queue entry identifies a job queue from which jobs are selected for running in the subsystem. Jobs started from a job queue are batch jobs. You add a job queue entry using the character-based interface.

#### Changing job queue entries

You can change an existing job queue entry in the specified subsystem description. This command can be issued while a subsystem is active or inactive. To change the job queue entry in a subsystem, use the character-based interface.

#### Removing job queue entries

You can remove job queue entries from a subsystem description by using the character-based interface. Jobs on the job queue remain on the queue when the job queue entry is removed from the subsystem description. A job queue entry cannot be removed if any currently active jobs were started from the job queue.

#### *Prestart job entries*

You define the prestart job by using a prestart job entry. A prestart job entry does not affect the device allocation or program start request assignment.

The job attributes of a prestart job are not changed by the subsystem when a program start request attaches to the prestart job. However, server jobs generally change job attributes to those of the swapped user profile.

The Change Prestart Job (CHGPJ) command allows the prestart job to change some of the job attributes to those of the job description (specified in the job description associated with the user profile of the program start request or in the job description specified in the prestart job entry).

#### *Prestart jobs for servers*

In the prestart job model there is one primary listening job, generally called the daemon job or listener job, and multiple server jobs that process the client requests. The daemon job listens on the port for connection requests. When a new connection is received, the daemon does some general work, then gives the socket descriptor to a waiting prestart server job.

Prestart jobs can be reused. When the job has completed the work for one client, the environment is reset and the job is made available to handle a request from a different client.

For server jobs that run user code (for example, the remote command server), the job typically is not reused. This is because the user code might have changed something in the job and there is no sure way to reset the environment for a new client. If your server does reuse the job, the Change Job (QWTCHGJB) API can be used to change the job's attribute back to a known state after the client's request has completed.

Servers that use the prestart job model include the host servers, SMTP server, PPP servers, DDM/DRDA Server, the SQL Server, and others.

### **Related concepts**

#### Prestart job investigation



This topic provides steps to help you answer the question, "How do I find the real user of a prestart job and determine the resources used by that prestart job?"

### **Related information**

[Experience Report: Tuning prestart job entries](#)

#### *Workstation entries*

An interactive job is a job that starts when a user signs on to a display station and ends when the user signs off. For the job to run, the subsystem searches for the job description, which may be specified in the workstation entry or the user profile.

The workstation entry guides the subsystem to prospective workstations. If a workstation is available, the subsystem sends a sign-on screen to the display.

**Note:** The subsystem description for the controlling subsystem must contain a workstation entry for the console, and that entry must be of type \*SIGNON. (\*SIGNON is a value for the AT parameter, specified on the Add Work Station Entry (ADDWSE) command.) The \*SIGNON value indicates that the sign-on display is shown at the workstation when the subsystem is started. This requirement ensures that the subsystem has an interactive device for the entry of the system and subsystem level commands. The End System (ENDSYS) command ends the IBM i licensed program to a single session (or sign-on display) at the console in the controlling subsystem. A subsystem description that does not contain a workstation entry for the console cannot be started as a controlling subsystem.

### **Related tasks**

#### Adding workstation entries

A workstation entry is used when a job is started when a user signs on or transfers an interactive job from another subsystem. You can specify the following items in a workstation entry. Parameter names are given in parentheses. Use the character-based interface to add workstation entries.

#### Changing workstation entries

You can specify a different job description for a previously defined workstation entry by using the character-based interface.

#### Removing workstation entries

You can remove a workstation entry from a subsystem description by using the character-based interface. The subsystem can be active at the time the command is run. However, all jobs that are active through the workstation entry must be ended before it can be removed.

### **Routing entries**

The routing entry identifies the main storage subsystem pool to use, the controlling program to run (typically the system-supplied program QCMD), and additional run-time information (stored in the class object). Routing entries are stored in the subsystem description.

A routing entry can be likened to a single entry in a shopping mall directory. Customers that cannot find the store they need may use a directory to help send them in the right direction. The same is true on your system. Routing entries guide the job to the correct place. Routing entries in a subsystem description specify the program to be called to control a routing step for a job running in the subsystem, which memory pool the job uses, and from which class to get the run-time attributes. Routing data identifies a routing entry for the job to use. Together, routing entries and routing data provide information about starting a job in a subsystem.

Routing entries consist of these parts; the subsystem description, class, comparison data, maximum active routing steps, memory pool ID, program to call, thread resources affinity, resources affinity group, and the sequence number.

### **Related tasks**

#### Adding routing entries

Each routing entry specifies the parameters used to start a routing step for a job. Routing entries identify the main storage subsystem pool to use, the controlling program to run (typically the system-supplied program QCMD), and additional run-time information (stored in the class object). To add a routing entry to a subsystem description, use the character-based interface.

### Changing routing entries

You can change a routing entry in the specified subsystem description by using the character-based interface. The routing entry specifies the parameters used to start a routing step for a job. The associated subsystem can be active when the changes are made.

### Removing routing entries

You can remove a routing entry from the specified subsystem description by using the character-based interface. The subsystem can be active at the time the command is run. However, the routing entry cannot be removed if there are any currently active jobs that were started using the entry.

### *Class*

Job run-time attributes are contained in the class object that is specified in the (CLS) parameter in the routing entry. If a job consists of multiple routing steps, the class used by each subsequent routing step is specified in the routing entry used to start the routing step. If the class does not exist when the routing entry is added, a library qualifier must be specified because the qualified class name is kept in the subsystem description.

Run-time attributes that are included in a routing entry class are:

#### **Run priority (RUNPTY)**

The run priority is a value ranging from 1 (highest priority) through 99 (lowest priority) that represents the priority at which the job competes for the processing unit relative to other jobs that are active at the same time. For multi-threaded jobs, the run priority is also the highest run priority allowed for any thread within the job. Individual threads within the job may have a lower priority.

#### **Time slice (TIMESLICE)**

This is the time slice establishes the amount of time needed by a thread in a job to accomplish a meaningful amount of processing. At the end of the time slice, the thread might be put in an inactive state so that other threads can become active in the storage pool.

#### **Default wait time (DFTWAIT)**

This specifies the default maximum time (in seconds) that a thread in the job waits for a system instruction, such as the LOCK machine interface (MI) instruction, to acquire a resource. This default wait time is used when a wait time is not otherwise specified for a given situation. Normally, this is the amount of time the system user might be willing to wait for the system before the request is ended. If the wait time for any one instruction is exceeded, an error message can be displayed or it can be automatically handled by a Monitor Message (MONMSG) command.

#### **Maximum CPU time (CPUTIME)**

This specifies the maximum processing unit time (in milliseconds) that the job can use. If the job consists of multiple routing steps, each routing step is allowed to use this amount of processing unit time. If the maximum time is exceeded, the job is held.

#### **Maximum temporary storage (MAXTMPSTG)**

This specifies the maximum amount of temporary (auxiliary) storage that the job can use. If the job consists of multiple routing steps, this is the maximum temporary storage that the routing step can use. This temporary storage is used for storage required by the program itself and by implicitly created internal system objects used to support the job. It does not include storage in the QTEMP library. If the maximum temporary storage is exceeded, the job is held. This parameter does not apply to the use of permanent storage, which is controlled through the user profile.

#### **Maximum threads (MAXTHD)**

This specifies the maximum number of threads that a job using this class can run with at any time. If multiple threads are initiated simultaneously, this value may be exceeded. If this maximum value is exceeded, the excess threads will be allowed to run to their normal completion. Initiation of additional threads will be inhibited until the maximum number of threads in the job drops below this maximum value.

#### **Text description (TEXT)**

This specifies the text that briefly describes the object. This is an attribute of the class object when it is created, but it is not a run-time attribute for a job.

## **Authority (AUT)**

This specifies the authority you are giving to users who do not have specific authority for the object, who are not on an authorization list, and whose group profile or supplemental group profiles do not have specific authority for the object. This is an attribute of the class object when it is created, but it is not a run-time attribute for a job.

### *Comparison data*

The comparison value (CMPVAL) parameter of the routing entry specifies data that is compared with routing data to determine which routing entry to use. (The routing entry also specifies the starting position for the comparison.) The routing data is compared with the comparison value of each routing entry in sequence number order until a match is found. The sequence number contained in a routing entry defines the order in which the routing entries are scanned and can be used as the identifier of the routing entry.

When a routing entry is found with a comparison value that matches the routing data, a routing step is started and the program specified in the routing entry is called. The run-time attributes in the class associated with the routing entry are used for the routing step, and the routing step runs in the storage pool specified in the routing entry.

You can specify a comparison value of \*ANY on the highest numbered routing entry. \*ANY means that a match is forced regardless of the routing data. Only one routing entry can contain the comparison value of \*ANY, and it must be the last (highest sequence number) entry in the subsystem description.

### *Maximum active routing steps*

The maximum active routing steps (MAXACT) parameter of the routing entry specifies the maximum number of routing steps (jobs) that can be active at the same time through this routing entry.

In a job, only one routing step is active at a time. When a subsystem is active and the maximum number of routing steps is reached, any subsequent attempt to start a routing step through this routing entry fails. The job that attempted to start the routing step is ended, and a message is sent by the subsystem to the job's log.

Typically there is no reason to control the number of routing steps, thus the recommended value is \*NOMAX.

### *Memory pool ID*

The memory pool ID (POOLID) parameter of the routing entry specifies the pool identifier of the storage pool in which the program runs. The pool identifier specified here relates to the storage pools in the subsystem description.

## **Program to call**

The program to call (PGM) parameter of the routing entry specifies the name and library of the program called as the first program run in the routing step. No parameters can be passed to the specified program. The program name can be either explicitly specified in the routing entry, or extracted from the routing data.

If a program name is specified in a routing entry, selection of that routing entry results in the routing entry program being called (regardless of the program name passed in an EVOKE function). If the program specified in the EVOKE function is supposed to be called, \*RTGDTA must be specified in this parameter. If the program does not exist when the routing entry is added or changed, a library qualifier must be specified because the qualified program name is kept in the subsystem description.

## **Sequence number**

The sequence number (SEQNBR) parameter of the routing entry tells the subsystem the order in which routing entries are to be searched for a routing data match. The routing entries are searched in sequence number order. When you add routing entries to a subsystem description, you should order them so that the entries likely to be compared most often are first. This reduces the search time.

Sequence Number	Comparison Value
10	'ABC'
20	'AB'
30	'A'
40	'E'
50	'D'

In the above example, the routing entries are searched in sequence number order. If the routing data is 'A', the search ends with routing entry 30. If the routing data is 'AB', the search ends with routing entry 20. If the routing data is 'ABC', the search ends with routing entry 10. Because routing data can be longer than the comparison value of the routing entry, the comparison (which is done in left-to-right order) stops when it reaches the end of the comparison value. Therefore, if the routing data is 'ABCD', the search ends with routing entry 10.

When you define routing entries, they must be ordered from the most specific to the most general. The following example shows a correct and incorrect way to define routing entries:

Correct		Incorrect	
Sequence Number	Comparison Value	Sequence Number	Comparison Value
10	'ABC'	10	'ABC'
20	'AB'	20	'ABCD'
30	'A'		
40	'E'		
9999	*ANY		

In the incorrect example, it is no longer possible to match routing entry 20 because any routing data that matches the comparison value for routing entry 20 matches the routing entry 10 first. When a routing entry is changed or added to a subsystem description with a comparison value that causes this situation, the system sends a diagnostic message identifying the situation.

The program named in the routing entry is given control when the routing step for the job is started. Parameters to control the run-time environment (priority, time slice, and so on) of the routing step for the job are taken from the class specified in the routing entry.

## How a 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.** The Start Subsystem (STRSBS) command is issued. Key startup information is located in the subsystem description.
- 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 storage pool is less than the minimum size specified by the Base memory pool minimum size system value QBASPOOL. 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.
- 3. Prestart jobs are started.** This information comes from the prestart job entries.
- 4. Autostart jobs are started.** This information comes from the autostart jobs entries.

5. **Display stations are allocated (sign-on displays are up).** 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 QCMNARB jobs hold locks on all varied-off devices. Workstation entries provide the information about what devices to check for allocation.

**Note:** For virtual display devices, the sign-on display is shown when the device becomes fully varied on. This happens when a user connects to the IBM i using that device description (assuming the connection request does not carry the data that is used to bypass the sign-on display processing). A device can be taken from a pool of previously created device descriptions and varied on as part of that connection processing, or a device can be created and varied on. At a subsystem start, the subsystem pends a lock for any of the previously created device descriptions that the subsystem wants.

6. **Job queues are allocated.** The subsystem will not be able to allocate a job queue if it is already allocated to another active subsystem. This information comes from the job queue entries.

7. **Communications devices are allocated.** Requests are sent to the QLU (LU services) system job, which handles device allocation for all communications devices. This information comes from the communication entries.

8. **The Environment is ready for work.**

### Related tasks

#### Starting a subsystem

The Start Subsystem (STRSBS) command starts a subsystem using the subsystem description specified in the command. When the subsystem is started, the system allocates the necessary and available resources (storage, workstations, and job queues) that are specified in the subsystem description. You can start a subsystem by using IBM Navigator for i interface or the character-based interface.

### ***How workstation devices are allocated***

Subsystems attempt to allocate all workstation devices in its subsystem description for AT(\*SIGNON) workstation entries.

The following situations might occur during the time the subsystem starts:

- 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.
- If 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 more than one subsystem tries to allocate the same workstation (as specified in the workstation entries) and the workstation is varied off, the subsystem that gets the workstation when it is varied on cannot be predicted. Similarly, if a workstation entry specifies a workstation type instead of a workstation name, a subsystem might get all, some, or none of the workstations of that type. (This also applies to workstation entries with generic names.) To avoid such a situation, you can set up the workstation entries for the subsystems so multiple subsystems are not using the same workstations.

### **After a user has signed on**

When a user signs on to a workstation, the job runs in the subsystem that was shown on the sign-on display on the workstation (the subsystem is identified in the IBM-supplied sign-on display). The following situations might occur after the user has signed on:

- If a second subsystem is started and it tries to allocate the workstation on which the user signed on, the second subsystem cannot allocate it. The user's job continues to run in the first subsystem.

- If the user selects option 1 (Display sign-on for alternative job) on the System Request menu or issues the Transfer to Secondary Job (TFRSECJOB) command, the new job runs in the same subsystem as the original job.
- When the user signs off, the workstation remains allocated to the subsystem used when the user signed on, unless the user transferred into the subsystem using the Transfer Job (TFRJOB) command, and specified AT (\*ENTER) for the workstation entry for this workstation. A sign-on display is shown, and any subsequent jobs from that workstation continue to run in that subsystem, (unless another subsystem is started up that allocates the workstation while it is at the sign-on display).
- If the user signs off and the subsystem in which his job was running is ended, the device is deallocated. A second subsystem can then allocate the device and display the sign-on display.

### Related tasks

#### Assigning users to a specific subsystem

You can use several techniques to assign device names and then associate those device names with users. After this is accomplished, you can use the workstation entries to get the user to the correct subsystem.

### Related information

Experience Report: Subsystem Configuration

Using Telnet exit point programs

### **Scenario: Workstation allocation**

This example illustrates how two workstations are allocated to two different subsystems.

In this scenario, subsystem A and subsystem B have workstations DSP01 and DSP02 in their subsystem descriptions (the workstation entries specify AT(\*SIGNON)).

Device Name	Allocated to
DSP01	Subsystem A
DSP02	Subsystem A

Assume that both workstations are varied on when subsystem A is started.

Subsystem A allocates both workstations and shows the sign-on display on both. Even though subsystem A has the sign-on display shown on the workstations, they can be allocated by another subsystem or job; the workstation is then no longer available to subsystem A.

Device Name	Allocated to
DSP01	USER1
DSP02	Subsystem A

When a user (USER1) signs on to workstation DSP01, the device is allocated to USER1's job, which is running in subsystem A. Workstation DSP02 is still at the sign-on display. Thus it can be allocated by another subsystem or job. It is then no longer available to subsystem A.

Device Name	Allocated to
DSP01	USER1
DSP02	Subsystem B

Subsystem B is started. Because USER1 has signed on to workstation DSP01, subsystem B cannot allocate the device. Subsystem B requests allocation of the device when it becomes available. DSP02 is allocated to subsystem B because no one has signed on to it in subsystem A. Any jobs started on DSP02 run in subsystem B.

Device Name	Allocated to
DSP01	Subsystem A
DSP02	Subsystem B

USER1 signs off. Because the user job was running in subsystem A, that subsystem displays the sign-on display so that another user can sign on the workstation and run in subsystem A. If subsystem A is ended, workstation DSP01 is allocated by subsystem B (because it has an outstanding request to allocate the device.)

The name of the subsystem that currently has a workstation allocated appears in the upper right corner of the IBM-supplied sign-on display.

### Related tasks

[Assigning users to a specific subsystem](#)

You can use several techniques to assign device names and then associate those device names with users. After this is accomplished, you can use the workstation entries to get the user to the correct subsystem.

### Related information

[Using Telnet exit point programs](#)

## 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 your system, all main storage can be divided into logical allocations called memory pools. By default, the system manages the transfer of data and programs into memory pools.

The memory pool from which user jobs get their memory is always the same pool that limits their activity level. (The activity level of a memory pool is the number of threads that can be active at same time in a memory pool.) Exceptions to this are system jobs (such as Scpf, Qsysarb, and Qlus) that get their memory from the Base pool but use the machine pool activity level. Additionally, subsystem monitors get their memory from the first subsystem description pool, but it uses the machine pool activity level. This allows a subsystem monitor to always be able to run regardless of the activity level setting.

### Why use memory pools

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

Using shared memory pools allows the system to distribute jobs for interactive users across multiple subsystems while still allowing their jobs to run in the same memory pool.

Multiple pools in a subsystem help you to control the jobs' competition for system resources. The advantages of having multiple pools in a subsystem are that you can separate the amount of work done and the response time for these jobs. For example, during the day you may want interactive jobs to run with good response time. For better efficiency you can make the interactive pool larger. At night you might be running many batch jobs, so you make the batch pool larger.

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

### How data is handled in memory pools

If data is already in main storage, it can be referred to independently of the memory pool it is in. However, if the needed data does not exist in any memory pool, it is brought into the same memory pool for the job that referred to it (this is known as a page fault). As data is transferred into a memory pool, other data is displaced and, if changed, is automatically recorded in auxiliary storage (this is called paging). The memory pool size should be large enough to keep data transfers (paging) at a reasonable level as the rate affects performance.

## Related concepts

### [Managing memory pools](#)

Making sure that jobs get enough memory to complete efficiently is important. If too much memory is given to subsystem A and not enough to subsystem B, jobs in subsystem B might begin to run poorly. The following information describes the various tasks that are involved in managing memory pools.

## Related information

### [Retrieve System Status \(QWCRSSTS\) API](#)

### [Managing system performance](#)

### [Basic performance tuning](#)

### [Applications for performance management](#)

### [Experience report: The Performance Adjuster \(QPFRADJ\)](#)

## Types of memory pools

On the your system, all main storage can be divided into logical allocations called *memory pools*. All memory pools in a system are either private or shared. There are private memory pools, shared memory pools, and special shared memory pools. As many as 64 memory pools, in any combination of private and shared pools, can be active at the same time.

### Private memory pools

*Private memory pools* (also known as user-defined memory pools) contain a specific amount of main storage that can be used by a single subsystem to run jobs. These pools cannot be shared by multiple subsystems. They are identified in IBM Navigator for i by the subsystem name. You can have as many as 62 private memory pools allocated for use in active subsystems.

### Shared memory pools

*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. 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).

#### Special Shared Pools (\*MACHINE and \*BASE)

##### \*MACHINE

The Machine memory pool is used for highly-shared Machine and operating system programs. It is identified as Machine in IBM Navigator for i. The Machine memory pool provides storage for tasks the system must run that do not require your attention. The size for this memory pool is specified in the Machine memory pool size system value (QMCHPOOL). No user jobs run in this memory pool. (On the Work with System Status display (WRKSYSSTS), the Machine memory pool appears as system pool identifier 1.)

##### \*BASE

The Base memory pool, identified as Base in IBM Navigator for i, contains all unassigned main storage on the system, (all main storage that is not required by another memory pool). The Base pool contains storage that can be shared by many subsystems. The Base memory pool is used for batch work and miscellaneous system functions. The Base memory pool minimum size (QBASPOOL) system value specifies the minimum size of the Base memory pool. The activity level for this memory pool is specified in the Base memory pool maximum eligible threads (QBASACTLVL) system value. (On the Work with System Status display (WRKSYSSTS), the Base memory pool appears as system pool identifier 2.)

#### General Shared Pools

*General shared pools* are pools of main storage that multiple subsystems can use at the same time. On the character-based interface, they are identified as follows:

- \*INTERACT is the interactive storage pool used for interactive jobs.



- \*SPOOL is the storage pool used for spool writers.
- \*SHRPOOL1 through \*SHRPOOL60 are storage pools that you can use for your own use.

In IBM Navigator for i, the general shared pools are identified as Interactive, Spool, and Shared 1 - Shared 60.

### Related tasks

#### [Creating a private memory pool](#)

Private memory pools (also known as user-defined memory pools) can be used by IBM-supplied subsystems or by user-defined subsystems. You can define up to a maximum of 10 memory pool definitions for a subsystem. You create a private memory pool in the subsystem description.

### Related information

#### [Managing system performance](#)

#### [Basic performance tuning](#)

#### [Applications for performance management](#)

#### [Experience report: The Performance Adjuster \(QPFRADJ\)](#)

#### [Performance system values: Machine memory pool size](#)

#### [Performance system values: Base memory pool minimum size](#)

#### [Performance system values: Base memory pool maximum eligible threads](#)

## Pool numbering schemes

Pools have two sets of numbering schemes: one is used within a subsystem and one is system-wide. The subsystem uses a set of numbers that refer to the pools it uses. Thus, when you create or change a subsystem description you can define one or more pools and label them 1, 2, 3, and so on. These are the designations of the subsystem pools, and they do not correspond to the pool numbers shown on the Work with System Status (WRKSYSSTS) display.

A different set of numbers is used to keep track of all pools on the system. The Work with Subsystems (WRKSBS) display relates the subsystem pool identifiers and the column headings to the system pool identifiers.

```

Work with Subsystems                               System: XXXXXXXX
Type options, press Enter.
 4=End subsystem 5=Display subsystem description
 8=Work with subsystem jobs

      Total      -----Subsystem Pools-----
Opt Subsystem Storage (M) 1 2 3 4 5 6 7 8 9 10
-  NYSBS             .48 2 4 5
-  PASBS             .97 2 6 5
-  QINTER            11.71 2 3

Bottom

Parameters or command
===>
F3=Exit F5=Refresh F11=Display system data F12=Cancel
F14=Work with system status

```

### Example: How pools are numbered

The following example illustrates how pools are numbered.

Subsystems		
CRTSBSD QINTER	CRTSBSD NYSBS	CRTSBSD PASBS
Pools (1 *BASE)	Pools (1 *BASE)	Pools (1 *BASE)

Subsystems		
(2 1200 25)	(2 500 3)	(2 1000 3)
(System pools 2, 3)	(3 *SHRPOOL2)	(3 *SHRPOOL2)
	(System pools 2, 4, 5)	(System pools 2, 5, 6)

After QINTER starts, the following pools are allocated:

System Pool Number	Description	QINTER
1	*Machine pool	
2	*BASE pool	1
3	QINTER private pool	2

After NYSBS starts the following pools are allocated:

System Pool Number	Description	QINTER	NYSBS
1	*MACHINE pool		
2	*BASE pool	1	1
3	QINTER private pool	2	
4	NYSBS private pool		2
5	*SHRPOOL2 shared pool		3

After PASBS starts the following pools are allocated:

System Pool Number	Description	QINTER	NYSBS	PASBS
1	*MACHINE pool			
2	*BASE pool	1	1	1
3	QINTER private pool	2		
4	NYSBS private pool		2	
5	SHRPOOL2 shared pool		3	3
6	PASBS private pool			2

### Related tasks

#### Managing tuning parameters for shared pools

To manage tuning parameters for shared pools, use IBM Navigator for i or character-based interface commands.

#### Managing a pool's configuration

To change a pool's size, activity level or paging option, use IBM Navigator for i or character-based interface commands.

#### Changing memory pool size

The size of a memory pool directly affects the amount of work that a subsystem can process. The more memory a subsystem has, the more work it can potentially complete. It is important that you monitor your system carefully before you start changing the parameters of your memory pools. You also want to periodically recheck these levels, as some readjustment might need to be done.

## Related information

[Managing system performance](#)

[Basic performance tuning](#)

[Applications for performance management](#)

[Experience report: The Performance Adjuster \(QPFRADJ\)](#)

## Memory pool allocation

When you start a subsystem, the system attempts to allocate the user-defined storage pools that are defined in the subsystem description of the started subsystem.

If the system cannot allocate all of the requested storage, it will allocate as much storage as is available and then allocate the remainder of the storage as it becomes available. For example, consider the following table. If 700KB is available, and if \*SHRPOOL2 is defined to 500KB, then 300KB is allocated to the first storage pool and 400KB is allocated to the second storage pool.

Pool ID Specified in SBSDB	1	2
Storage Requested	300K	*SHRPOOL2
System Pool ID	3	4
Storage Allocated	300K	400K
Activity Level	1	
Pool Type	Private	Shared

The storage pools that you define decrease the size of the Base memory pool when they are allocated. The system only allocates as much storage to a private pool as it has available in the Base memory pool. The Base memory pool minimum size (QBASPOOL) system value determines the minimum Base pool size.

## Related tasks

[Viewing memory pool information](#)

You can view information about the memory pools that are on your system by using IBM Navigator for i or the character-based interface.

[Determining the number of subsystems using a memory pool](#)

Subsystems are allocated a certain percentage of memory to run jobs. It is important to know how many different subsystems are pulling from the same memory pool. After you know how many subsystems are submitting jobs to a pool and how many jobs are running in a pool, you might want to reduce resource contention by adjusting the size and activity level of the pool.

[Determining the number of jobs in a memory pool](#)

IBM Navigator for i provides you with a way to quickly display a list of jobs that are currently running in a memory pool.

[Determining in which pool a single job is running](#)

If you have a job that is not performing as you expect you might want to check the memory pool in which the job is running. To determine in which pool a single job is running, use IBM Navigator for i or the character based interface.

## Related information

[Managing system performance](#)

[Basic performance tuning](#)

[Applications for performance management](#)

[Experience report: The Performance Adjuster \(QPFRADJ\)](#)

## Memory pool activity level

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. This allows for efficient use of system resources. The system manages the control of the activity 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 the excess threads must 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 typically 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 that you specify.

### Maximum activity level

After 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.

### Data memory pools

A shared memory pool defined with an activity level of zero is a data memory pool. No threads can run in the pool, it can only be used for data.

### Related tasks

#### Viewing memory pool information

You can view information about the memory pools that are on your system by using IBM Navigator for i or the character-based interface.

#### Determining the number of subsystems using a memory pool

Subsystems are allocated a certain percentage of memory to run jobs. It is important to know how many different subsystems are pulling from the same memory pool. After you know how many subsystems are submitting jobs to a pool and how many jobs are running in a pool, you might want to reduce resource contention by adjusting the size and activity level of the pool.

#### Determining the number of jobs in a memory pool

IBM Navigator for i provides you with a way to quickly display a list of jobs that are currently running in a memory pool.

#### Determining in which pool a single job is running

If you have a job that is not performing as you expect you might want to check the memory pool in which the job is running. To determine in which pool a single job is running, use IBM Navigator for i or the character based interface.

### **Related information**

[Managing system performance](#)

[Basic performance tuning](#)

[Applications for performance management](#)

[Experience report: The Performance Adjuster \(QPFRADJ\)](#)

## **Jobs**

All work done on a system is performed through jobs. Each job has a unique name within the system. All jobs, with the exception of system jobs, run within subsystems. A job can enter the subsystem from any of the work entries, such as a job queue entry, workstation entry, communications entry, autostart job entry, or prestart job entry.

Each active job contains at least one thread (the initial thread) and may contain additional secondary threads. Threads are independent units of work. Job attributes are shared among the threads of the job, however threads also have some of their own attributes, such as a call stack. The job's attributes contain information about how the work is processed. The job serves as the owner for attributes 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 attributes.

### **Proper authority**

To make most changes to a job's attributes, you need to have job control special authority (\*JOBCTL) or your user profile matches the job user identity of the job being changed.

There are a few attributes where having \*JOBCTL special authority is necessary to make any changes. These attributes are:

- Default wait time
- Run priority
- Time slice

**Note:** If you plan to make changes to the job's accounting code, you need \*USE authority to the Change Accounting Code (CHGACGCDE) command in addition to \*JOBCTL special authority or a user profile matching the job's job user identity.

For any job attributes that refer to an IBM i object, such as job queues, output queues, and sort sequence tables, you need to have the proper authority to the object. For more details about IBM i authorities, see [Authority required for objects used by commands](#) in the Security reference topic collection.

### **Related concepts**

[Job user identity](#)

The *job user identity (JUID)* is the name of the user profile by which this job is known to other jobs. This name is used for authorization checks when other jobs attempt to operate against this job.

## **Job characteristics**

Work management provides a way for you to control the work done on your system through a job's attributes. However, before you can control the various aspects of a job, you need to understand the different characteristics of a job.

The following information describes the characteristics of jobs:

## **Job name syntax**

To make it easier to control and identify jobs on the system, each job has a unique qualified job name. The qualified job name consists of three parts: the job name (or simple job name), the user name, and the job number.

- For interactive jobs, the job name is the same as the name of the workstation or emulator session that you signed on to. For batch jobs you can specify your own job name. The job name can be up to 10 characters long.
- The user name is the name of the user profile under which the job is started. For interactive jobs, the user name is the user profile used to sign on to the system. This is the user name that you entered in the user field on the sign-on display. If you are using Telnet and by-passing the sign-on, this is the user name that you use to automatically sign on to the system. For batch jobs you can specify the user profile under which the batch job is to run. The user name can be up to 10 characters long.
- The job number is a unique number assigned by the system so that you can identify jobs, even if more than one has the same job name and user name. The job number is always 6 numeric digits.

## **Syntax**

The syntax for qualified job names is similar to qualified names for objects. For example, if the job name is DSP01, the user is QPGMR, and the job number is 000578, the qualified job name is entered on the Work with Job (WRKJOB) command as follows:

```
WRKJOB JOB(000578/QPGMR/DSP01)
```

Another similarity to object names is that you do not need to specify all of the qualifiers. For example consider the following:

```
WRKJOB JOB(QPGMR/DSP01)
```

or

```
WRKJOB JOB(DSP01)
```

This works the same as entering the entire qualified job name. If several jobs on the system match the portion of the job name that you entered, the Select Job display appears. This display allows you to select which job you want from a list of duplicate job names.

## **Job Attributes**

Job attributes determine how the system runs each job. Some job attributes are set from the user profile. Other job attributes come from system values, from locales, from a Submit Job (SBMJOB) command, from a job description, and from the Change Job (CHGJOB) command (from which you can change values for attributes while the job is running).

Controlling job attributes gives you the flexibility to control jobs at the job level, user level, or system level. For example, you can have your system set up to go all the way to the system value to get job attributes (which is the system default). Then if you want to change a value for all new jobs on the system, you can change the system value.

By specifying a value in a job description, you can affect all of the types of jobs that use that job description. For example, if all of your batch jobs use the same job description, then changing the job description for the batch jobs can affect all of your batch jobs and leave all other jobs unaffected.

## **Related information**

[Experience report: Work management job attributes](#)

## **Job description**

The job description allows you to create a set of job attributes that are saved and available for multiple uses. The job description can be used as the source for some of the job attributes that tell the system how to run a job. The attributes tell the system when to start the job, where to get the job from, and how the

job will run. You can think of a job description as a template that many jobs can use, thereby reducing the number of specific parameters that you need to set for each individual job.

Job descriptions are used by autostart, batch, interactive, and prestart job types. You can use the same job description for multiple jobs. When you define a job, you can use the job description in one of two ways:

- Use a specified job description without overriding any of its attributes. For example:

```
SBMJOB JOB(OEDAILY) JOBD(QBATCH)
```

- Use a specified job description but override some of the attributes (using BCHJOB or SBMJOB command). For example, to override the message logging in the job description QBATCH, specify:

```
SBMJOB JOB(OEDAILY) JOBD(QBATCH)  
LOG(2 20 *SECLVL)
```

**Note:** You cannot override any job description attributes for autostart jobs, workstation jobs, or communication jobs.

### Related tasks

#### Creating a job description

You can use the character-based interface, the Work With Job Description (WRKJOB) command or the Create Job Description (CRTJOB) command to create job descriptions.

#### Using a job description

The most common way to use a job description is by specifying it in the Submit Job (SBMJOB) command. The job description (JOB) parameter is where you specify the job description that you want this job to use. When you define a batch job, you can use the job description in one of two ways:

### Job descriptions and security

Every job in the system uses a job description during job initiation. This controls the various attributes of a job. The USER parameter controls the name of the user profile assigned to the job. A job description that has a user profile name (USER) specified should be authorized only to specific individuals. If not, at security level 30 and below, other users will be able to submit jobs to run under that user profile.

For example, consider

```
CRTJOB JOB(XX) USER(JONES) . . . AUT(*USE)
```

This example has security risks because any user can submit a job using the XX job description, and be authorized to whatever JONES is authorized to. If this type of job description is used on a workstation entry, it allows anyone to sign on as that user just by pressing the Enter key. To avoid any security exposure, do not authorize this type of job description to \*PUBLIC.

**Note:** At security level 40 and 50, the Submit Job (SBMJOB) command requires the submitter to be authorized (\*USE) to the user profile named in the job description. This assumes that the SBMJOB specifies user (\*JOB). Nevertheless, avoid specifying a user in a job description unless it is needed for some specific reason (such as an autostart job) and you tightly control access to it.

### USER Parameter and Interactive Jobs

The job description to be used is defined on the Add Work Station Entry (ADDWSE) command. The default is to use the job description in the user profile. If USER(\*RQD) is specified in the job description, the user must enter a user name. If USER(XXXX) is specified (where XXXX is a specific user profile name), the user is allowed to press the Enter key on the sign-on display and operate under the XXXX user profile name, unless the security level is 40 or higher.

### USER Parameter and Batch Jobs

The job description used for batch jobs is specified on the Submit Job (SBMJOB) or Batch Job (BCHJOB) command.

If an input stream is entered that contains the BCHJOB command, the user entering one of the Start Reader commands ( STRDBRDR, STRDKTRDR) or one of the Submit Job commands (SBMDBJOB, SBMDKTJOB, and so on.) must have object operational authority (\*OBJOPR) to the job description that is specified. When an input stream is used, jobs always operate under the user profile of the job description and not of the user who is placing the jobs on the job queue. If USR(\*RQD) is specified in the job description, it is invalid to use the job description on a BCHJOB command.

If a SBMJOB command is used, the command defaults so that the batch job operates under the user profile name of the submitter. However, if USER(\*JOBID) is specified on the SBMJOB command, the job operates under the name specified in the USER parameter of the job description.

Frequently a specific name in the job description is required to let users submit work for a specific user profile. For example, the QBATCH job description is shipped with USER(QPGMR) to allow this. To avoid any security exposure, do not authorize this type of job description to \*PUBLIC.

### **Call stacks**

The *call stack* is the ordered list of all programs or procedures currently running for a job. The programs and procedures can be started explicitly with the CALL instruction, or implicitly from some other event.

The call stack is available at both the job level and the thread level. On the character-based interface, the call stack is a last-in-first-out (LIFO) list of call stack entries, one entry for each called procedure or program. In IBM Navigator for i, by default, the last entry in the stack appears at the top of the list. However, the ordering can be changed by using the **Sort ascending** or **Sort descending** buttons.

The information that is included in the Call Stack display includes the invocation information for the original program model (OPM), integrated language environment (ILE), IBM i Portable Application Solutions Environment (PASE), and Java™ applications. Also, if you are running under a user profile with \*SERVICE special authority, you can see additional entries for licensed internal code (LIC) and IBM i PASE Kernel.

### **Related tasks**

#### Viewing call stacks

You can view information about a job or thread's call stack by using either IBM Navigator for i or the character-based interface.

### **Class object**

A class object contains the run attributes that control the run-time environment of a job. IBM-supplied class objects, or classes, meet the needs of both typical interactive and batch applications. The following classes (by name) are supplied with the system:

- QGPL/QBATCH: For use by batch jobs
- QSYS/QCTL: For use by the controlling subsystem
- QGPL/QINTER: For use by interactive jobs
- QGPL/QPGMR: For use by the programming subsystem
- QGPL/QSPL: For use by the spooling subsystem printer writer
- QGPL/QSPL2: For general spooling use in the Base system pool

### **Run-time attributes**

The following is a list of some of the run-time attributes, or parameters, that are found in a class object that are important to work management.

#### **Run priority (RUNPTY)**

A number that specifies the priority level assigned to all jobs running that use the class. The priority level is used to determine which job, of all the jobs competing for system resources, is run next. The value can be 1 through 99, where 1 is the highest priority (all jobs having a 1 priority are run first). This value is the highest run priority allowed for any thread within the job. Individual threads within the job may have a lower priority. Changing the run priority of the job affects the run priorities of all threads within the job. For example, if the job is running at priority 10, thread A within the job is running at



priority 10 and thread B within the job is running at priority 15. If the priority of the job is changed to 20, then the priority of thread A is adjusted to 20 and the priority of thread B is adjusted to 25.

### **Time slice (TIMESLICE)**

The maximum amount of processor time (in milliseconds) given to each thread in a job using this class before other threads in a job or other jobs are given the opportunity to run. The time slice establishes the amount of time needed by a thread in a job to accomplish a meaningful amount of processing. At the end of the time slice, the thread might be put in an inactive state so that other threads can become active in the storage pool.

### **Default wait time (DFTWAIT)**

The default amount of time that the system waits for the completion of an instruction that performs a wait. This wait time applies to times when an instruction is waiting for a system action, not to the time an instruction is waiting for a response from a user. Normally, this is the amount of time you are willing to wait for the system before ending the request. If the wait time is exceeded, an error message is passed to the job. This default wait time is used when a wait time is not otherwise specified for a given situation.

The wait time used for allocating file resources is specified in the file description and can be overridden by an override command. It specifies that the wait time specified in the class object is used. If file resources are not available when the file is opened, the system waits for them until the wait time ends.

**Note:** The class attributes apply to each routing step of a job. Most jobs have only one routing step, but if the job is rerouted (because of something like the Reroute Job (RRTJOB) or Transfer Job (TFRJOB) command) the class attributes are reset.

### **Maximum CPU time (CPUTIME)**

The maximum amount of processor time allowed for a job's routing step to complete processing. If the job's routing step is not completed in this amount of time, it is held, and a message is written to the job log.

### **Maximum temporary storage (MAXTMPSTG)**

The maximum amount of temporary storage that can be used by a job's routing step. This temporary storage is used for the programs that run in the job, for the system objects used to support the job, and for temporary objects created by the job.

### **Maximum threads (MAXTHD)**

The maximum number of threads in which a job in this class can run at any time. If multiple threads are initiated simultaneously, this value may be exceeded. The excess threads are allowed to run their normal completion. Initiation of additional threads are inhibited until the maximum number of threads in the job drops below this maximum value.

**Note:** The resources used by the threads and the resources available on the system can vary. Therefore, the initiation of additional threads may be inhibited before this maximum value is reached.

### **Related tasks**

#### Creating a class object

You can create a class object by using the character-based interface. The class defines the processing attributes for jobs that use the class. The class used by a job is specified in the subsystem description routing entry used to start the job. If a job consists of multiple routing steps, the class used by each subsequent routing step is specified in the routing entry used to start the routing step.

#### Changing a class object

You can change the attributes of a class object by using the character-based interface. Any attribute can be changed, except for the public authority attribute. Refer to the Revoke Object Authority (RVKOBJAUT)

command and the Grant Object Authority (GRTOBJAUT) command for more information about changing object authorizations.

### **Job user identity**

The *job user identity (JUID)* is the name of the user profile by which this job is known to other jobs. This name is used for authorization checks when other jobs attempt to operate against this job.

Some examples of functions that operate against another job include the Start Service Job (STRSRVJOB) command, the Retrieve Job Information (QUSRJOB) API, the Change Job (QWTCHGJB) API, all job control commands, and functions that send signals from one job to another.

In situations where jobs swap user profiles, the current user profile identifies the profile under which the initial thread is running instead of the JUID.

The JUID is not used to make authorization checks from within a job. Authorization to perform a function is always based on the current user profile of the thread in which the function is called.

When a job is on a job queue or output queue, the JUID is always the same as the user name of the job and cannot be changed.

When a job starts, and at the start of any subsequent routing steps, the JUID is the same as the name of the current user profile of the job. While a job is active, the JUID can be changed in the following ways.

- The JUID can be explicitly set by an application using the Set Job User Identify (QWTSJUID) application program interface (API) or the QwtSetJuid() function. The JUID is set with the name of the user profile that the thread that called the API or function is running under.
- The JUID can be explicitly cleared by an application using the QWTSJUID API or the QwtClearJuid() function. The job must be running as a single threaded job at the time. When cleared, the JUID is implicitly set by the system to the name of the user profile that the single thread of the job is running under at that point.
- If the job is running as a single threaded job, and the JUID has not been explicitly set by an application, then each time the job uses the Set Profile (QWTSETP) API to run under a different user profile the JUID is implicitly set by the system to the name of the user profile that was set by QWTSETP.
- When a single threaded job initiates a secondary thread and the JUID has not been explicitly set by an application, then the system will implicitly set the JUID with the name of the user profile that the single thread of the job was running under at the point that it initiated the secondary thread.

When the job returns to a single thread, the system implicitly sets the JUID to the name of the user profile that the single thread of the job is running under at that point.

### **Related concepts**

#### Proper authority

To make most changes to a job's attributes, you need to have job control special authority (\*JOBCTL) or your user profile matches the job user identity of the job being changed.

### **Job user identity examples**

These examples illustrate how the job user identity (JUID) is assigned in different situations.

- A job runs under a user profile called USERA. The JUID is USERA. If the job uses the QWTSETP API to switch to USERB, the JUID changes to USERB.

In this situation, the Set By value for the JUID is \*DEFAULT. Because the job that is running single-threaded, the job user identity is the current user profile under which the initial thread of the job is running (unless, the job user identity was explicitly set by an application). For job queue jobs and completed jobs, the job user identity is the user name from the qualified job name.

- A single-threaded job runs under user profile USERX. The JUID is USERX. If the job initiates secondary threads, the JUID remains as USERX. If all the threads then swap to USERY, the JUID is still USERX.

In this situation, the Set By value for the JUID is \*SYSTEM. Because this is an active job which is currently running as a multi-threaded job, the job user identity is implicitly set

by the system. The job user identity is set to the name of the user profile under which the job was running when the job became multi-threaded. When the job returns to running single-threaded, the job user identity will be reset to the \*DEFAULT value.

- If a server running under a user profile called SERVER calls the QWTSJUID API, the JUID will be set to SERVER. If the server then calls the Set Profile (QWTSETP) API to set its current user profile to CLIENT while processing work on behalf of that client, the JUID remains as SERVER. Likewise, if the server initiates secondary threads that each call QWTSETP to run under various user profiles, the JUID remains as SERVER.

In this situation, the Set By value for the JUID is \*APPLICATION. The job user identity is set explicitly by an application using an API. This value applies to both single-threaded and multi-threaded jobs.

## **Threads**

The term *thread* is shorthand for "thread of control". A thread is the path taken by a program while running, the steps performed, and the order in which the steps are performed. A thread runs code from its starting location in an ordered, predefined sequence for a given set of inputs.

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

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. In the threads on IBM Navigator for i, 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.

## **Thread types**

The thread type determines how the thread was created on the system.

### **User**

The thread can be created by the customer application. The initial thread in a job is always a user thread. The Allow multiple threads field 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.

## **Related tasks**

### Viewing 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.

### Viewing threads running under a specific job

Every active job running on your 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.

### Ending or deleting threads

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 might not be able to complete without that thread's work.

## **Related information**

Example: End a thread using Java

Thread management APIs

### Proper thread authority

Certain authority levels are required before you can work with threads.

To view and change most attributes of a thread you need to have \*JOBCTL special authority, or your user profile needs to match the job user identity of the job containing the thread. To change the run priority of a thread, you must have \*JOBCTL special authority. Thread Control authority allows you to view some of the attributes of a thread.

To hold or release a thread, you need to have \*JOBCTL special authority or Thread Control authority, or your user profile needs to match the job user identity of the job containing the thread. To end a thread, you need to have \*SERVICE special authority or Thread Control authority.

For any thread attributes that refer to a IBM i object, such as a library in the library list, the user needs to have the proper authority to the object.

For more details about IBM i authorities, see [Authority required for objects used by commands](#) in the Security reference topic collection.

**Note:** With thread Control authority, you can retrieve information about threads of another job. Thread Control can be granted and revoked for individual users by using IBM Navigator for i 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 about application administration, see the Information Center topic Application Administration.

### Thread status

The current status of a thread is viewed from the General page in the Thread Properties window, under Detailed status.

Value number	Status	Character-based interface value
1	Running	RUN
2	Job held	HLD
3	Held	HLDT
4	Stopped by a signal	SIGS
5	Waiting for save while active checkpoint	CMTW
6	Waiting for condition	CNDW
7	Waiting for dequeue	DEQA/DEQW
8	Waiting for event	EVTW
9	Waiting for activity level	INEL
10	Waiting for Java program	JVAA/JVAW
11	Waiting for lock	LCKW
12	Waiting for lock space	LSPW
13	Waiting for mutex	MTXW
14	Waiting for select	SELW
15	Waiting for semaphore	SEMW
16	Waiting for signal	SIGW
17	Waiting for thread	THDW

<i>Table 1. Thread status values (continued)</i>		
<b>Value number</b>	<b>Status</b>	<b>Character-based interface value</b>
18	Waiting for time interval	TIMA/TIMW
19	Unknown	Blank

**Note:** In properties, threads that have been held more than once will have Held (n) status, where n is the number of times the thread has been held. For performance reasons, the held count will not be displayed in the Detailed Status column. For threads that are waiting for a lock, a dequeue, or a lock space, additional information is provided that identifies the item being waited on.

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 IBM i 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, 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 about the different thread statuses, see the IBM Navigator for i online help.

### **Locked objects**

Jobs and threads use objects to process work.

Because more than one piece of work is processing at a time, a lock is put on an object so that data integrity is retained. *Locked objects* are system objects used by jobs and threads to process work. After the job or thread is finished running, the object is unlocked and ready to be used to process more work. Depending on the lock request type used, locking an object permits only one user to use an object at a time. For example, if two or more users tried to change an object at the same time, the changes to the object by the second user is locked out until the first user finished updating the object. With the use of lock holders, a user can see what currently has a lock or is currently waiting on a lock for an object.

*Scope* specifies whether the lock is associated with a job, a thread, or a lock space. Scope also defines how long the lock will be available and what lock request type and conflict rules the object has on it.

*Lock request types* are different levels of access that a job, thread or lock space can use to an object that is locked. For example, a lock exclusive, no read lock type is used if an object is being changed or deleted on the system. This lock request type does not allow anyone to use the object, nor does it allow anyone to read the object.

The different lock request types are:

### **Exclusive - No read**

The object is reserved for exclusive use. However, if the object is locked by any lock request type, you cannot obtain exclusive use of the object. This lock state is appropriate when a user does not want any other user to have access to the object until the function being performed is complete.

### **Exclusive-Read**

The object can only be shared with the shared-read lock request type. This lock is appropriate when a user wants to prevent other users from performing any operation other than a read.

### **Shared-Update**

The object can be shared with either the shared-read or shared-update lock request type. That is, another user can request either a shared-read lock state or a shared-update lock state for the same object. This lock state is appropriate when a user intends to change an object but wants to allow other users to read or change the same object.

### **Shared-No update**

The object can be shared with only share - no update, and shared-read lock request types. This lock state is appropriate when a user does not intend to change an object but wants to ensure that no other user changes the object.

### **Shared-Read**

The object can be shared with all lock requests other than exclusive - no read. That is, another user can request an exclusive-read, shared-update, shared-read, or shared-no update lock state.

The *lock status* tells the state of the lock request. The different lock statuses are:

**Held:** The lock request has been fulfilled and the job, thread or lock space is holding the lock.

**Waiting:** The job or thread is waiting to obtain the lock.

**Requested:** The job or thread has requested the lock.

*Lock holders* are the jobs, threads and lock spaces that are currently holding a lock or are waiting for a lock on a specific locked object.

## **Job types**

Your system processes several different types of jobs. This information describes those jobs and how they are used.

### **Autostart jobs**

An autostart job is a batch job doing repetitive work, one-time initialization work that is associated with a particular subsystem, initializes functions for an application, or provides centralized service functions for other jobs in the same subsystem. An autostart job in the controlling subsystem can be used to start other subsystems (as does the IBM-supplied controlling subsystem). The autostart jobs associated with a subsystem are automatically started each time the subsystem is started.

»To add an autostart job entry to the subsystem description, use the Add Autostart Job Entry (ADDAJE) command. When the subsystem starts the autostart job, it uses the job description specified in the autostart job entry to determine the routing data (RTGDTA) and tries to match that to the compare value in the routing entries defined in the subsystem description. When a match is found, other information in the routing entry is used to establish the work environment for the job such as the class that will be used and the program that will be run. When the program name in the routing entry is QCMD, the command processing program will look for the request data or command (RQSDTA) on the job message queue. The request data is found in the job description for the autostart job entry.«

If more than one autostart job is specified for a subsystem, all autostart jobs are started immediately rather than one followed by another. If the value specified for the maximum jobs in subsystem (MAXJOBS) is exceeded, no other jobs can be started in the subsystem until enough autostart jobs have completed so that the number of jobs running is below the maximum jobs.

The job description that is used for an autostart job is specified using the Add Autostart Job Entry (ADDAJE) command. When the subsystem is started, the job operates under the user profile name in the specified job description. You may not specify the job description which contains USER(\*RQD). Because

the autostart job operates under the user profile that is specified by the job description, you need to control who is allowed to change the job description.

### **Batch jobs**

A batch job is a predefined group of processing actions submitted to the system to be performed with little or no interaction between the user and the system. Jobs that do not require user interaction to run can be processed as batch jobs. A batch job typically is a low priority job and can require a special system environment in which to run.

Batch jobs run in the system background, freeing the user who submitted the job to do other work. Several batch jobs can be active at the same time.

The following list describes the different kinds of batch jobs:

#### **Simple batch job**

The simple batch job is a job that is submitted to a job queue. It waits in line with other batch jobs and is processed according to its priority and sequence number.

#### **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. (This is accomplished by using the `spawn()` API.) 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.

Batch jobs can be started when a user:

- Causes a job to be placed in a job queue
- Issues a communication program start request
- Starts a subsystem with a prestart job
- Uses the `spawn()` API

#### *How a batch job starts*

When a user submits a batch job, the job gathers information from several system objects before it is placed on a job queue.

1. A user submits a job.
2. The job searches for job attributes. If the job attributes are not found on the Submit Job (SBMJOB) command, the job looks in the job description (specified on the SBJOB command), the current user's user profile, and the currently active job (the job issuing the SBJOB command).

**Note:** Similar to interactive job initiation, you can specify in the job description to use the user profile. The user profile can specify to use a system value to find certain job attributes.

3. Once the job has all of its attributes, it resides on the job queue.
4. When the subsystem is ready to handle a job, it looks for jobs in the job queues (those that the subsystem has allocated).
5. Then, like interactive job processing, the subsystem checks the job description for the routing data.
6. The subsystem uses the routing data to find a routing entry. The routing entry provides information about which pool the job uses, which routing program is used, and from which class the job gets its run-time attributes.
7. After this information is obtained, the routing program is run. If you use QCMD, QCMD carries out the SBJOB command. It runs the command specified on the CMD or RQSDTA parameter.

## Related tasks

### Submitting a batch job

Since batch jobs are typically low priority jobs that require a special system environment in which to run (such as running at night) they are placed in batch job queues. In the job queue the batch job receives a run time schedule and a priority. To submit a job to a batch job queue, you use the character-based interface and one of two commands.

### Starting a batch job that is waiting in the job queue

Occasionally you might need to force a job to start immediately. While moving the job to a job queue that is not busy is the most efficient method to accomplish this, there are some other methods that you can use.

## Related information

### QPRTJOB job

### *Spawn batch jobs*

*Spawn* is a function that creates a new job process (child process) that inherits many attributes of the calling process (parent process). A new program is specified and starts running in the child process. When you spawn a batch job you are using a parent job to pass along arguments and environment variables to the child job. The `spawn()` API uses batch immediate jobs, prestart jobs, or prestart batch jobs.

## Related information

### spawn()--Spawn Process

SPAWN CL command, QUSRTOOL example

## Communication jobs

A communications job is a batch job that is started by a program start request from a remote system. Job processing involves a communication request and appropriate specifications.

For a communications batch job to run on an IBM i system, a subsystem description containing a work entry for communications jobs must exist on the system. The communications work entry identifies to the subsystem the sources for the communications job it processes. The job processing begins when the subsystem receives a communications program start request from a remote system and an appropriate routing entry is found for the request.

## Routing data for communication jobs

Job routing of communications jobs is determined by the program start request that is received from the remote system. When a program start request is processed on the target system, a fixed-length data stream that is used as routing data is created. Position 25 of the routing data always contains PGMEVOKE for communications requests. Subsystem routing entries that specify a compare value of PGMEVOKE in position 29 typically have \*RTGDTA as the program name. This means that the program name specified in the routing data (from the remote system's program start request) is the program to run.

If a special processing environment is required for certain communications jobs, you can add an additional routing entry to the subsystem description, specifying a compare value whose starting position is 37. This compare value should contain the program name for the program start request. The routing entry must have a sequence number lower than the routing entry that uses PGMEVOKE as the compare value. This method allows certain communications jobs to run with a different class or pool specification.

## Security

The security on the system controls who can use communications devices as well as who can access the commands used with the associated device descriptions. You should consider additional security measures when writing and running application programs on both remote and target systems.

## Job description for communication jobs

The job description used for communications jobs is specified on the Add Communications Entry (ADDCMNE) command. The user specified on this job description is ignored. The system gets the user



name for communications jobs from the program start request. If the program start request does not specify a user name, the system uses the default user value from the communications entry. To ensure a greater degree of system security, include user information about the program start request rather than specifying a default user in the communications work entry.

### ***Interactive jobs***

An interactive job is a job that starts when a user signs on to a display station and ends when the user signs off. For the job to run, the subsystem searches for the job description, which can be specified in the workstation entry or the user profile.

Interactive jobs require continual two-way communications between the user and the system to perform a task. An interactive job begins when a user signs on to 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, or the job ends due to an application exception or device error recovery.

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.

**Did you know?** You can sign on to the system in two ways. You can manually enter the system by using a user id and password. You can also create a program to automatically send the user id and password to the server, thereby bypassing the sign-on screen.

#### *How an interactive job starts*

When a user signs on to the system, the subsystem gathers information from several system objects before the interactive job is ready.

1. The subsystem looks in the workstation entry for the job description in order to get the attributes for the interactive job. If the workstation entry specifies \*USRPRF for the job description, the job uses the information from the user profile.  
**Note:** This flexibility allows you to specify whether the job's attributes are tied to the workstation or to the individual user.
2. After the subsystem knows which job description to use, it might not find all of the job attributes in the job description. Some attributes might be in the user profile. If the user profile does not have the information, the subsystem looks at the system value.  
**Note:** The user profile contains job attributes that allow you to tailor certain things specifically for that user.
3. After the subsystem gathers all of the job's attributes, it determines whether a new interactive job can start or if an error message should be posted on the sign-on screen. The subsystem checks whether the maximum number of jobs allowed by the subsystem or by the workstation entry has been reached. Then it verifies that a valid user profile name has been supplied, that the user profile name is an enabled user profile, and that the supplied password (if required) is valid. Next, it verifies that the user has the proper authorities to the job description, the subsystem description, the workstation device description, and the output queue and library. Finally, the subsystem checks whether the user has reached the limits for allowed sign-ons for that user profile. If any validation errors are encountered, the sign-on screen displays with an appropriate message. Otherwise, the process of starting the interactive job continues.
4. After the subsystem validates that the interactive job can start, it checks the job description for the routing data. The subsystem uses the routing data to find a routing entry in the subsystem description. The routing entry provides information about which pool the job uses, which routing program is used, and from which class the job gets its run time attributes.

5. When all of these pieces are obtained, the routing program runs. IBM supplies a routing program called QCMD, which you can use for all types of work. QCMD knows if the job is an interactive job and checks the user profile for an initial program to run. If the initial program finishes running, QCMD displays the initial menu.

### **Related tasks**

#### Avoiding a long-running function from a workstation

To avoid a long-running function (such as save/restore) from a workstation without tying it up, the system operator can submit the job to a job queue.

#### *Disconnecting interactive jobs*

When the Disconnect Job (DSCJOB) command is called, the job is disconnected and the sign-on display is shown again. To connect with the job again, sign on to the same device from which you disconnected. Another interactive job may be started on the device under a different user name.

- An option on the System Request menu allows you to disconnect an interactive job, causing the sign-on display to appear. The option calls the DSCJOB command.
- When connecting with a job again, the values specified on the sign-on display for program, menu, and current library are ignored.
- A job which has PC organizer or PC text assist function active cannot be disconnected.
- All jobs are disconnected for group jobs. When they are connected again, you return to the place where the disconnect was issued. If the last active group job ends before you connect again, you return to the next group job.
- If the job cannot be disconnected for any reason, the job is ended instead.
- All disconnected jobs in the subsystem end when the subsystem ends. If a subsystem is ending, the DSCJOB command cannot be issued in any of the jobs in the subsystem.
- The Disconnect Job Interval (QDSCJOBITV) system value can be used to indicate a time interval for which a job can be disconnected. If the time interval is reached, the disconnected job ends.
- Disconnected jobs that have not exceeded the QDSCJOBITV value end when the subsystem is ended or when an IPL occurs.

### **Related concepts**

#### Job disconnection considerations

There are several factors that you must consider whenever you disconnect a job.

### **Related tasks**

#### Ending interactive jobs

You can use several different methods to end an interactive job.

#### Disconnecting all jobs from a device

The Disconnect Job (DSCJOB) command allows the interactive user to disconnect all interactive jobs at the workstation and return to the sign-on display. The switched line is dropped only if that is specified in the workstation device description of this workstation and if no other workstation on this line is active. If the job is disconnected when the disconnect interval in the Time-out interval for disconnected jobs (QDSCJOBITV) system value is reached, the job is ended and the job log is not included with the job's spooled output.

#### *I/O error for job requester device*

A requester device is a workstation from which a user can log on to a domain and use network resources. The Device Recovery Action (DEVRCYACN) job attribute specifies what action to take when an I/O error occurs for a job's requester device.

The DEVRCYACN attribute has the following options:

#### **\*SYSVAL**

This is the default. It points to the Action to take when a device error occurs on the workstation (QDEVRCYACN) system value. The system value supports all of the values that the job attribute supports (except \*SYSVAL).

**\*MSG**

Signals the I/O error message and lets the application program perform error recovery. This is NOT the recommended setting.

**\*DSCMSG**

Disconnect the job. This is the shipped default. Upon connecting again, a new error message signals the user's application program indicating the device was lost and recovered since the I/O, and the contents of the display need to be shown again.

**\*DSCENDRQS**

Disconnect the job. Upon connecting again, an end request function is performed to return control of the job to the last request level.

**\*ENDJOB**

End the job. A job log might be produced for the job. A message is sent to the job log and to the QHST log indicating the job ended because of the device error.

**\*ENDJOBNOLOG**

End the job. No job log is produced. A message is sent to the QHST log indicating the job ended because of the device error.

**Note:** If \*DSCENDRQS, \*ENDJOB, or \*ENDJOBNOLOG is specified for DEVRCYACN, the recovery action takes effect when the error occurs on the device. If one of the other values is specified, the recovery action takes place at the next I/O to the device with the error.

*Interactive jobs and routing steps*

Before the initial menu is called the routing data is compared with the routing entries in the subsystem description. When a match is made, the program specified in the routing entry is called and the routing step is started.

The following illustrates the subsequent activity leading up to starting a routing step and displaying the initial menu for a user profile specifying an initial program.

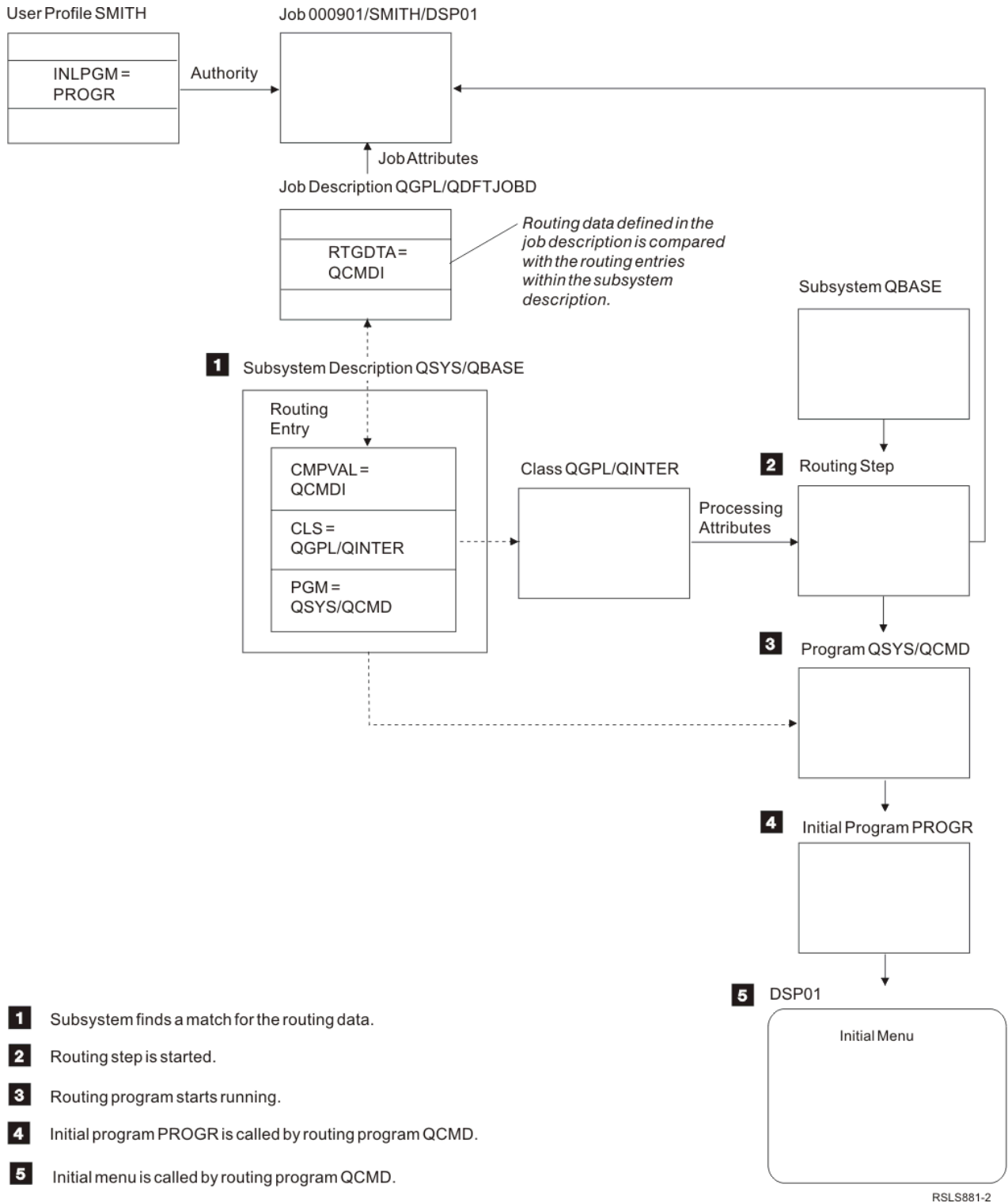


Figure 1. Subsystem Activity

## Interactive Job Approaches

You can handle interactive jobs in various ways. These approaches are dependent upon how you control the routing step. First you should determine the following:

- Which program will control the routing step: QSYS/QCMD or a user program?
- Will the routing be user-based or workstation-based?

### *Programs that control the routing step*

To determine the best approach for a particular job, you must first determine which program should control the routing step.

## **Using QSYS/CMD for interactive jobs - benefits**

The IBM-supplied command processor QSYS/QCMD gives the greatest flexibility in terms of making functions available to workstation users. Using QCMD to control the routing step gives you the following benefits:

- The attention program is activated if it is specified in the user profile.
- The initial program that is specified in the user profile is called.
- The initial menu that is specified in the user profile is called.
- The user is placed in System/36 environment as it is specified in the user profile.

In addition, the default using QCMD brings you to the Main Menu where you can enter commands directly, including the CALL command, which is used to call user-written functions. Menu options with online help are provided to give easy access to system functions. Also provided are command selection menus, quick access to index search, and the command entry function (called by CALL QCMD). The command entry functions are intended primarily for programmers and operators who require the full range of functions available through the direct use of commands.

## **Calling a user program directly for interactive jobs - benefits**

Your programs can be directly called to control the routing steps for interactive jobs. These programs can be designed to give a more specialized access to functions needed by your workstation users than the IBM-supplied programs give. In addition, because your programs are tailored for specific functions, they should typically require even less system resource to support their running than the IBM-supplied programs. You may also want to provide functions such as an initial program and initial menu.

### *Workstation versus user based routing*

After you have determined which program controls the routing step, you must determine if routing is to be based on the workstation from which the job was started, or on the user (user profile) who signed on.

Routing based on the workstation is accomplished using the routing data specified in the job description associated with the workstation entry or profile for the device. Routing based on a user can be done using the initial program specified in the user profile or the job description in the user profile mapping to a routing entry other than QCMD.

## **Initial program uses**

Initial programs may interact with workstations to get input values from a workstation user. When an initial program is called, it cannot receive parameter values. An initial program can be used in one of two ways:

- To establish an initial environment for the user entering commands. For example, the library list can be changed or print files and message files can be overridden. When an initial program completes its function and returns to QSYS/QCMD, the initial menu is displayed.
- As the controlling program for the job. If the initial program does not return to QSYS/QCMD, it becomes the controlling program for the routing step. The initial menu is not displayed. The user can only request those functions available through the initial program.

For example, a menu can be displayed with specific application options. The user can only perform the functions on the menu. One example of such an option is sign off. If the SIGNOFF command is run, the job ends and the system Main Menu is never displayed. If you use this approach, consider using the user profile option INLMNU to ensure that no menu is displayed.

An initial program can be written so that when a return is issued, it either does or does not return to QSYS/QCMD. If the initial program returns to QSYS/QCMD, the initial menu is displayed.

### *When jobs end at the same time*

Sometimes, jobs end at the same time. For example, a network error occurs and the job attributes are set to \*ENDJOB or \*ENDJOBNO LIST. In addition to the job ending, the following device recovery actions occur.

- The job's priority is lowered. This occurs so the job is no longer at the same priority as the other active interactive jobs.
- The job's time slice is set to 100 milliseconds. This occurs to give higher priority jobs a better chance of getting processing resources.

Job logs for jobs with job attributes set to \*ENDJOB or \*ENDJOBNO LIST are in job log pending. To produce printer output from a job log that is in job log pending, use the Display Job Log (DSPJOBLOG) command.

When a job ends you can control how the job log is written to a spooled file. This can be done by the job itself while it is ending, by a background server job, or not at all. The value that you specify can have a significant impact on overall recovery time when many jobs end at the same time. For more information, see the related concept Job log pending.

### **Related concepts**

#### Job log pending

The job log pending state has been available for many years. When the job log attribute of a job is \*PND, no job log is produced. You can control how and under what circumstances the job log for a specific job is produced.

### ***Prestart jobs***

A prestart job is a batch job that starts running before a work request is received. The prestart jobs are started before any other types of jobs in a subsystem. Prestart jobs are different from other jobs because they use prestart job entries (part of the subsystem description) to determine which program, class, and storage pool to use when they are started.

Within a prestart job entry, you must specify attributes that the subsystem uses to create and manage a pool of prestart jobs. Use prestart jobs to reduce the amount of time required to handle a work request. 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.

### **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.

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 attributes 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 attributes 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. Prestart jobs provide the ability to initialize once and handle many requests so that a new job is not needed for every request. Many 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.

## Program Start Requests

A Program Start Request (PSR) is an architected way for SNA clients to connect to an SNA server. When a prestart job is set up to handle PSRs, the external state of the job is in PSRW (Program Start Request Wait).

Prestart jobs are also used for IBM-supplied TCP/IP servers, most notably the host servers. These prestart jobs accept work via internal interfaces and PSRs are not used. However, prestart jobs that are waiting for work, even if they are not using PSRs, still show a PSRW state.

### Related concepts

[Prestart communications jobs and job accounting](#)

If your system uses job accounting, the prestart job program should run the Change Prestart Job (CHGPJ) command with the program start request value for the accounting code parameter (CHGPJ ACGCDE(\*PGMSTRRQS)) immediately after the program start request attaches to the prestart job.

### Related tasks

[Starting a prestart job](#)

Prestart jobs typically start at the same time the subsystem is started. You manually start a prestart job when all prestart jobs have been ended by the system due to an error or were never started during subsystem start up due to STRJOBS (\*NO) on the prestart job entry. To start a prestart job, use the character-based interface.

[Ending a prestart job](#)

You can use the character-based interface to end a prestart job in an active subsystem.

### Related information

[Experience Report: Tuning prestart job entries](#)

#### *Prestart job name*

The fully qualified three-part name of the prestart job never changes once a prestart job is started. The user name of the fully qualified three-part job name always contains the user profile under which the prestart job is started.

If a spooled file is opened before a prestart job handles any work request, the spooled file is associated with the prestart job entry user profile. Otherwise, it is associated with the current user profile of the job.

If the prestart job entry profile and the current user profile are different, spooled files are spooled under a job with the job name being QPRTJOB and the user name of the current user profile. (This is also true for prestart job entries for server jobs.)

The class (CLS) parameter on the prestart job entry provides a way to control the performance characteristics of two classes of prestart jobs per prestart job entry.

#### *How prestart jobs work*

A prestart job is a job that is started before the work arrives. This allows the system to handle a request for work without the delay caused by starting a new job.

A prestart job is a unique type of batch job. This means that the job has a job type of 'B' and a job subtype of 'J'. The enhanced job type further defines the job as a prestart job (1610), prestart batch job (1620), or prestart communications job (1630). The enhanced job type describes how the prestart job accepts work requests. If the program running in the prestart job uses the communications interface for accepting work, the job is a prestart communications job. If the program running in the prestart job accepts work through a batch work interface, the job is a prestart batch job. If the program has not yet reached the point of accepting work, the job is just a prestart job. Prestart batch jobs are often referred to as server jobs because they provide service for the work requests.

A communications work request is handled by the subsystem that has the required communications device allocated. A batch work request is typically handled by one of the basic subsystems that are shipped with the system: QSYSWRK, QUSRWRK, or QSERVER.

Prestart jobs are started based on the information contained in the prestart job entries. The Start jobs (STRJOBS) parameter of the Add Prestart Job Entry (ADDPJE) and the Change Prestart Job Entry

(CHGPJE) commands can specify that the prestart jobs are started when the subsystem is started or when the Start Prestart Jobs (STRPJ) command is entered. The Initial number of jobs (INLJOBS) parameter determines the number of prestart jobs that initially start for a program.

As work requests arrive, more prestart jobs may be needed. The Threshold (THRESHOLD) parameter of the Add Prestart Job Entry (ADDPJE) and the Change Prestart Job Entry (CHGPJE) commands tells when to start more jobs. When the number of prestart jobs available to handle a request drops below the value specified by the THRESHOLD parameter, the additional jobs are started. The Additional number of jobs (ADLJOBS) parameter tells how many more jobs to start.

Some prestart jobs handle a work request and then become available to handle another work request. The Maximum number of uses (MAXUSE) parameter allows you to specify how many work requests these prestart jobs handle. Some prestart jobs handle a single work request and then end, ignoring the MAXUSE value. Whether the prestart job handles multiple work requests or handles only a single work request is determined by the program running in the prestart job.

When the prestart job ends after handling at least one work request, the subsystem compares the number of jobs that are still running to the number specified in the INLJOBS parameter. If the number of jobs remaining is less than INLJOBS, the subsystem starts another job.

If a prestart job ends without handling at least one work request and the job was not ended by the End Job (ENDJOB) command, the prestart job program is considered to be in error. The subsystem ends the prestart job entry in a controlled manner. This allows jobs that are servicing a work request to complete that request, but prevents the subsystem from starting additional jobs.

The subsystem periodically checks the number of prestart jobs to determine if there are excessive available prestart jobs. A prestart job is available when it is waiting for a work request.

### **Related information**

[Experience Report: Tuning prestart job entries](#)

[Experience Report: Subsystem Configuration](#)

#### *Prestart job entries*

You define the prestart job by using a prestart job entry. A prestart job entry does not affect the device allocation or program start request assignment.

The job attributes of a prestart job are not changed by the subsystem when a program start request attaches to the prestart job. However, server jobs generally change job attributes to those of the swapped user profile.

The Change Prestart Job (CHGPJ) command allows the prestart job to change some of the job attributes to those of the job description (specified in the job description associated with the user profile of the program start request or in the job description specified in the prestart job entry).

### **Related concepts**

[Prestart job investigation](#)

This topic provides steps to help you answer the question, "How do I find the real user of a prestart job and determine the resources used by that prestart job?"

### **Related tasks**

[Adding prestart job entries](#)

Prestart job entries identify prestart jobs that may be started when the subsystem is started or when the Start Prestart Jobs (STRPJ) command is entered. You can add prestart job entries to the subsystem description by using the character-based interface.

[Changing prestart entries](#)

You can change a prestart job entry in the specified subsystem description. The subsystem may be active when the prestart job entry is changed. Changes made to the entry when the subsystem is active are reflected over time. Any new prestart jobs started after the command is issued use the new job-related values. This command identifies prestart jobs that are started when the subsystem is started or when the Start Prestart Jobs (STRPJ) command is issued.

[Removing prestart job entries](#)



You can remove prestart job entries from the subsystem description by using the character-based interface. A prestart job entry cannot be removed if any currently active jobs were started using the entry.

### **Related information**

[Experience Report: Tuning prestart job entries](#)

#### *Prestart job handling program start requests*

When a prestart job starts, it runs under the prestart job user profile. When a program start request attaches to a prestart job, the prestart job user profile is replaced by the program start request user profile. When the prestart job is finished handling a program start request, the program start request user profile is replaced by the prestart job user profile. If there is a group profile associated with the user profile, the group profile is also swapped.

The swapped user profile is for authority checking only. None of the other attributes associated with the user profile are swapped. Libraries on the library list to which the prestart job entry user profile is authorized continue to be authorized to the prestart job when the program start request user profile replaces the prestart job entry user profile. However, the library list can be changed by the Change Library List (CHGLIBL) command.

### **Prestart job object authorization for program start requests**

When a prestart job starts, authority checking against the prestart job entry user profile is performed on every object that is needed for starting a job. Before a program start request is allowed to attach a prestart job, only the program start request user profile/password and its authority to the communications devices and library/program is checked.

To avoid occurrences where the program start request user profile is not authorized to objects that the prestart job entry user profile is authorized to, you must ensure that the program start request user profile is authorized to at least as many objects as the prestart job entry user profile. To accomplish this, the prestart job program can be created by the prestart job entry user with USRPRF(\*OWNER) specified on the CRTxxxPGM (where xxx is the program language) command. The program owner authority will automatically be transferred to any programs called by the prestart job program. Otherwise, you may choose to explicitly check object authorization (CHKOBJ) before referring to any objects.

Files and objects that the prestart job user profile is not authorized to should be closed and deallocated before the end of the transaction is performed on the requestor device. If database files are left open in the prestart job, in order to guarantee database security, the prestart job program must check the program start request user profile authority to the open files.

#### *Prestart jobs for batch applications*

Prestart jobs and server jobs that use prestart jobs present a unique situation for job accounting. If a single prestart job services different users you might want to charge each of these users for their resources used. In such a situation the accounting code needs to be updated before and after each service request.

For more information about how job accounting and prestart jobs relate, see [“Prestart communications jobs and job accounting”](#) on page 87.

#### *Performance tips for prestart jobs*

The prestart job should do as much work as possible before it attempts to acquire an ICF program device or accept a CPI Communications conversation. The more work it does initially (allocating objects, opening database files, and so on), the less it needs to do when a program start request is received, therefore giving the transaction faster response time. The following are some additional performance considerations when using prestart jobs:

**Remember:** If an active prestart job entry is in the subsystem, the subsystem periodically checks the number of prestart jobs in a pool that are ready to service program start requests to determine if there are excessive available prestart jobs. Excessive available prestart jobs are ended by the subsystem gradually. However, the subsystem always leaves at least the number of prestart jobs specified in the INLJOBS attribute in a pool.

- You should only deallocate resources specific to the transaction that you want performed. Any resource that is commonly used for other transactions performed by the prestart job program should remain allocated while the job is waiting for its next request. You should leave files open and objects allocated to save time when the next request is received.

**Note:** Database files that are left open in the prestart job generally require the same considerations as database files that are shared in the same job.

- Since the same QTEMP library is used for the entire life of a prestart job, objects that are no longer needed should be deleted.
- Since the same Local Data Area (LDA) is used for the entire life of a prestart job, information can be kept and passed to the next transaction.
- Since each prestart job can handle many program start requests, and has only one job log, you may want your application to send messages to the job log identifying the activity of the prestart job. This is also useful because the job logs of batch prestart jobs are cleared between uses.
- The job attributes of a prestart job are not changed by the subsystem when a program start request attaches to a prestart job. The change Prestart Job (CHGPJ) command allows the prestart job to change some of the job attributes to those of the job description (specified in the job description associated with the user profile of the program start request or in the job description specified in the prestart job entry.)
- The class (CLS) parameter on the prestart job entry provides a way to control the performance characteristics of two classes of prestart jobs per prestart job entry. For example, you can provide a lower execution priority for work that arrives when the system is already busy.

#### *Spooled file and the prestart job entry*

If a spooled file is opened before a prestart job handles any program start request, the spooled file is associated with the prestart job entry user profile; otherwise it is associated with the current program start request user profile.

If the prestart job entry profile and the current program start request user profile are different, spooled files are spooled under a job with the first part of the three-part job name being QPRTJOB and the second part being the name of the user profile.

### **Reader and writer jobs**

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

#### **Reader**

A reader job reads batch job streams from database 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.

### **Server jobs**

Server jobs are jobs that run continuously in the background on your system.

Work can come from network functions, operating system functions, on behalf of a user, another system within the network, or from general system services, such as the clustering server jobs. Server jobs typically run in one of three basic subsystems that are shipped with the system - QSYSWRK, QSERVER, or QUSRWRK. Server jobs are most commonly associated with such functions as HTTP, Lotus Notes, and TCP/IP. Your system has three basic models for server jobs:

#### **Threaded Job Model**

In the threaded job model the server job is a job with multiple threads. One thread acts as the distributor of work to the other threads. For example, when the server receives a client request, the initial thread reads the request and passes it to another thread to fulfill the request. With this model,

the amount of jobs on the system is greatly reduced because work is handled in different threads rather than requiring multiple jobs. A few examples of server jobs that use the threaded job model are Domino®, HTTP server, and WebSphere®.

### **Prestart Job Model**

In the prestart job model there is typically a primary job that acts as a listener for requests that come into the system. This job is typically called the daemon job. The daemon job handles the initial request and then passes the request to the appropriate prestart server job. With this job model, using prestart jobs can reduce the number of jobs that are required because after a request has been fulfilled the prestart server job waits for the next request. The server job is reused. Also, from a performance perspective, the prestart job is already running and waiting to process the request. Some examples of server jobs that use the prestart job model are SQL server, host servers, and Simple Mail Transfer Protocol (SMTP).

**Note:** For jobs that run user code, typically the job is not reused (like most server jobs). This is because the user code may have changed anything in the job (such as the remote command server).

### **Multiple Listening Job Model**

In the multiple listening job model, several server jobs are started. When a request comes in, the job that receives the request handles the job request, while the next available server job waits for the next request to come in. Once the server job completes the request, it closes the connection and ends. A new server job starts and the cycle continues.

With this model, you do not need to be concerned with prestart job entries. However, sometimes configuring subsystems unique to your environment is not possible because this model runs in the default subsystem. One exception is File Transfer Protocol (FTP). With file transfer protocol you can configure the subsystem in which the file transfer protocol server runs. There is no ability to have some FTP work to run in one subsystem and the rest of the work to run in a different subsystem. Also, from a performance perspective, the cost of job initiation and job termination cannot be avoided because once a job is run it is ended and another job starts. However, because jobs end when the connection is complete and the next job is started, the new job will generally be up and running when the next request is received, so the job initiation and termination cost should not affect the time it takes to connect to the server.

Some examples of server jobs that use the multiple listening job model are FTP and line printer daemon (LPD).

For more detailed information about the job names of the server jobs that run on the system, see the server job table. This table shows you the subsystem and the job name so that you can find the active job and its job logs. The table also shows the job description each server job uses. By default most server jobs do not generate a job log when the job ends (the LOG parameter is set to 4 0 \*NOLIST), which means that the job log is not created. If you want a job log to be generated with all the messages sent to the job log, the LOG parameter needs to specify 4 0 \*SECLVL.

### **Related information**

[Server job table](#)

### **System jobs**

System jobs are created by the operating system to control system resources and perform system functions. System jobs run when the server starts or when an independent disk pool is varied on. These jobs perform a variety of tasks from starting the operating system, to starting and ending subsystems, to scheduling jobs.

### **Related information**

[Cluster jobs](#)

### *System startup jobs*

*Startup jobs* are system jobs that run at IPL. They handle the tasks that get the operating system environment up and ready for work. The following is a list of the various system startup jobs.

#### **Scpf (start control program function)**

This is the central job when you start the system. Scpf starts the Qsysarb series, but Qsysarb3 starts most of the other system jobs (not 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 (Pwrdownsys) 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 typically 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.

#### **Qlpsvr (software agreements acceptance)**

This job is automatically started during an IPL if online software agreements need to be accepted. The job ends when all agreements are either accepted or declined.

### *System arbiters*

The system arbiters (QSYSARB and QSYSARB2 through QSYSARB5), started by an SCPF system job, provide the environment for the running of high-priority functions. They allow subsystems to start and end and keep track of the state of the system (for example, a restricted state).

The system arbiters, identified by the job name QSYSARB and QSYSARB2 through QSYSARB5, are the central and highest priority jobs within the operating system. Each system arbiter responds to system-wide events that must be handled immediately and those that can be handled more efficiently by a single job than multiple jobs.

The system arbiter (QSYSARB) is also responsible for starting the Logical Unit Services (QLUS) job during an IPL. The system arbiter remains active until the system is ended.

The following is a list of 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, Qtaparb (tape arbiter), 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. Qsysarb3 also starts and ends many of the system jobs.

#### **Qsysarb4 (system arbiter 4)**

This job is responsible for starting and ending subsystems. This includes the initial power down (Pwrdownsys) processing.

#### **Qsysarb5 (system arbiter 5)**

This job is responsible for processing machine events. This includes handling events to support auxiliary power, 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.

### *System communication jobs*

This topic contains a list of system communication jobs.

#### **Qlus (logical unit services)**

Qlus handles the event handling for logical unit devices, known as communication devices. Qlus is also responsible for allocating devices to the correct communications subsystem.

#### **Qcmnarbxx (communication arbiters)**

The communications arbiters with Qsysarb (system arbiter) and Qtaparb (tape arbiter) process work for all types of devices, not just communication devices. This work includes communications connection, disconnection, device locking, and error recovery processing. At restart, the system value communication arbiter jobs (QCMNARB) 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 these remote file systems.

### *Database jobs*

This information contains a list of database jobs.

#### **Qdbfstccol (database file statistic collection)**

This job collects database file statistics. These statistics are crucial to correct database query optimization.

#### **Qdbsrvxr (database cross-reference) and Qdbx####xr for independent disk pool group ###**

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) and Qdbx####xr2 for independent disk pool group ###**

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) and Qdbs####v01 for independent disk pool group ###**

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. Qdbsrv01 is the main system job assigning work to the others. Typically, Qdbsrv01 is 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. (Example: Qdbsrv02 through Qdbsrv05 are high priority, Qdbsrv06 through Qdbsrv09 are low priority.)

#### **Qdbsrvxx (database server, high priority) and Qdbs####vxx for independent disk pool group ###**

These jobs perform journal and commitment control maintenance for the system and are considered quick or short-running work.

**Qdbsrvxx (database server, low priority) and Qdbs####vxx for independent disk pool group ###**

These jobs perform access path maintenance on user data files. Typically, these jobs are inactive, but in certain cases, they might activate to perform access path rebuilds. Some reasons why these jobs might 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-iServer installation activity to complete cross-reference or other DB upgrade 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 system jobs*

This information contains a list of other kinds of system jobs.

**Qalert (alert manager)**

This job performs the tasks necessary to process alerts. 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)**

This job decompresses newly installed operating system objects as needed. There is a storage requirement in order for these jobs to run. If the 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 (files system)**

This job supports the background processing of the integrated file system. It ensures that changes to the 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.

**Qli####cl for independent disk pool group ### (library cleanup)**

This job cleans up libraries on independent disk pools.

**Qli####rp for independent disk pool group ### (object cleanup)**

This job cleans up replaced objects on independent disk pool libraries.

**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 Automatically adjust memory pools and activity levels (Qpfradj) system value 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) and Qspmn##### for independent disk pool group #####**

This job performs system spooling functions that include:

- Clears the spooled database member which contained a deleted spooled file's data and attributes
- Deletes the spooled database members that have not been reused within the time specified in Automatically clean up unused printer output storage (QRCLSPLSTG) system value

**Qsprc00001 (system spool recovery) and Qsprc##### for independent disk pool group #####**

This job performs system spooling functions that include:

- Spooled file cleanup after an IPL or an independent disk pool group is varied on
- Moves stranded spooled files of damaged user output queues into the output queue QSPRCLOUTQ in library QRCL or QRCL#####.

**Qspff00001 and Qspp200001 (system spool PRTQ updaters); Qspff##### and Qspp2##### for independent disk pool group #####**

These jobs perform spooled file operations for either the system disk pool or a specific independent disk pool group.

**Qtaparb (tape device)**

This job processes work related to tape devices including device locking and error recovery processing.

**Qnwharbx**

These system jobs handle events related to the Network Server Host Adapter (NWSH) devices. There will always be at least one of these jobs started during the current IPL.

**Qwcpjobs**

This job handles the background cleanup of permanent job structures.

**Qwctjobs**

This job handles the background cleanup of temporary job structures.

## Job scheduling options

The job schedule function allows for time-dependent scheduling of IBM i batch jobs. You can schedule jobs to be released from the job queue at a particular time, or you can use a job schedule entry to submit your job to the job queue automatically at the time you specify. Job scheduling allows you to control the date and time a batch job is submitted to or becomes eligible to start from a job queue. This flexibility can help you as you balance the work load on your system.

For example, you can use job scheduling to delegate the repetitive task of repeatedly submitting meeting notices, payroll, or weekly and monthly reports from your schedule to the system's schedule. There are four methods for scheduling a batch job.

### Management Central scheduler

System i® Navigator provides an integrated scheduler, the Management Central scheduler, to organize when you want your jobs to process. You have the option of choosing to perform a task immediately or choosing a later time. You can use the Management Central scheduler to schedule almost any task in Management Central.

The Management Central Scheduler window is available anytime you see a **Schedule** button on a System i Navigator window.

**Note:** If you installed the Advanced Job Scheduler on the Management Central server, the **Schedule** button will start the Advanced Job Scheduler instead of the Management Central scheduler.

**Related tasks**

[Scheduling a job using Management Central Scheduler](#)

If you do not have the plug-in Advanced Job Scheduler installed, you can use the Management Central Scheduler to schedule jobs.

### Job schedule entries

If your system does not have the Management Central Scheduler or the Advanced Job Scheduler, you can still schedule jobs using a job schedule entry, which is accessed from the character-based interface. Using this method you can schedule jobs to recur or to run only once.

Since job schedule entries are entries in a permanent object, they do not stay on the job queue like the scheduled jobs, and therefore they are not lost when the job queue is cleared. You can also save and restore the job schedule object. This provides a method of backing up your job scheduling information.

When you want a job to process at regular intervals, you create a job schedule entry for the job. The job schedule entry contains all of the information that is necessary to submit a job and its scheduling information. Each entry in the object is uniquely identified by the job name that you supply and a 6-digit entry number that is assigned by the system. No two entries have the same job name and entry number combinations.

The job schedule entry also contains information used by the system to manage the entry in certain situations. The information that defines the job is similar to the parameters specified on a Submit Job (SBMJOB) command, including job name, job description, job queue, user profile, and message queue. The local data area (LDA) of the job submitted from the job schedule entry is blank when the job starts.

All job schedule entries are contained in the job schedule object. The job schedule object, QDFTJOBSCD is in the QUSRSYS library and has an object type of \*JOBSCD. You cannot create, delete, rename, or duplicate the job schedule object. You cannot move it to any other library. The job schedule object is shipped with public authority of \*CHANGE. This is the minimum authority necessary to add, change, hold, release, and remove job schedule entries.

**Note:** You can also schedule recurring jobs by using the Management Central Scheduler or the Advanced Job Scheduler.

### Related concepts

#### Working with job schedule entries

In addition to the IBM Navigator for i Job Properties - Job Queue window, you can also change the job schedule entry directly by using the character-based interface. The following is a list of common character-based interface tasks that you can use when working with job schedule entries.

### **Examples: job schedule entry**

This topic provides examples for using the Add Job Schedule Entry (ADDJOBSCDE) command.

**Schedule a job monthly:** This example shows how to submit a job to run program INVENTORY at 11:30 p.m. on the last day of every month except on New Year's Eve.

```
ADDJOBSCDE JOB(MONTHEND)
CMD(CALL INVENTORY)
SCDDATE(*MONTHEND)
SCDTIME('23:30:00')
FRQ(*MONTHLY)
OMITDATE('12/31/05')
```

**Schedule a job daily:** This example shows how to submit a job to run program DAILYCLEAN every day at 6:00 p.m. The job runs under the user profile SOMEPMGR. This job is not submitted if the system is down or is in restricted state at that time.

```
ADDJOBSCDE JOB(*JOB)
CMD(CALL DAILYCLEAN)
SCDDAY(*ALL)
SCDTIME('18:00:00')
SCDDATE(*NONE)
USER(SOMEPMGR)
FRQ(*WEEKLY)
RCYACN(*NOSBM)
```

**Schedule a job weekly:** This example shows how to submit a job to run program PGM1 every week starting on 12/17/05 at the current time. Because 12/17/05 is a Saturday, the job is submitted every Saturday, and it runs under the user profile

```
PGMR1. ADDJOBSCDE JOB(*JOB)
CMD(CALL PGM1)
SCDDATE('12/17/05')
FRQ(*WEEKLY)
USER(PGMR1)
```

**Schedule a job every third Monday and Wednesday:** This example shows how to submit a job to run program PGM2 on the third Monday and the third Wednesday at 11:30 p.m. This job will be submitted on the next third Monday or third Wednesday at 11:30 p.m., depending on whether those days have passed



already this month. If yesterday was the third Monday, today is the third Tuesday, and tomorrow is the third Wednesday, it will be submitted tomorrow, and then not again until next month.

```
ADDJOBSCDE JOB(*JOB)  
CMD(CALL PGM2)  
SCDDAY(*MON *WED) FRQ(*MONTHLY)  
SCDDATE(*NONE)  
RELDAYMON(3) SCDTIME('23:30:00')
```

**Schedule a job every first and third Monday:** This example shows how to submit a job to run program PAYROLL on the first and third Monday of every month at 9:00 a.m. The job runs under user profile PAYROLLMGR.

```
ADDJOBSCDE JOB(PAYROLL)  
CMD(CALL PAYROLL)  
SCDDAY(*MON) FRQ(*MONTHLY)  
SCDDATE(*NONE)  
RELDAYMON(1 3) SCDTIME('09:00:00')  
USER(PAYROLLMGR)
```

**Schedule a job every weekday:** This example shows how to submit a job to run PGM4 every weekday at 7:00 p.m.

```
ADDJOBSCDE JOB(*JOB)  
CMD(CALL PGM4)  
SCDDAY(*MON *TUE *WED *THU *FRI)  
SCDDATE(*NONE)  
SCDTIME('19:00:00') FRQ(*WEEKLY)
```

**Save a job schedule entry:** This example shows how to submit a job once and save the entry.

```
ADDJOBSCDE JOB(*JOB)  
CMD(CALL SAVED)  
FRQ(*ONCE)  
SAVE(*YES)
```

## The submit job command

This character-based interface command controls the time a job is released in the job queue. It is an easy way to schedule a job that only needs to run once. It allows you to use many of the job attributes defined for your current job.

When you schedule a job to run only once (character-based command SBMJOB), the job is released from the job queue at the scheduled time. The following is a summary of the system tasks that occur when you use SBMJOB to schedule a batch job.

1. You schedule a job using either the IBM Navigator for i interface (**Basic Operations > User Jobs > Right-click the job > Properties > Job Queue tab**) or the character-based interface (SBMJOB command with SCDATE and SCDTIME parameters specified).
2. The job remains on the job queue in a scheduled state (SCD status) until the date and time indicated by the parameters.
3. At the scheduled time, the job is released from the job queue. The job's status changes from scheduled (SCD) to released (RLS), unless the job is held (SCDHLD), in which case it changes from scheduled to held (HLD).
4. The job is processed like any other job on the job queue.
5. The job starts if normal conditions (such as a job queue allocated to an active subsystem and maximum jobs not already active) exist.

**Note:** This method places the job on the job queue immediately, thus if the job queue is cleared before the scheduled date and time, you lose your job.

### Related tasks

[Submitting a job once](#)

When you need to run a job once, whether immediately or at a scheduled date and time, use the Submit Job (SBMJOB) command. This method places the job on the job queue immediately.

### Submitting a batch job

Since batch jobs are typically low priority jobs that require a special system environment in which to run (such as running at night) they are placed in batch job queues. In the job queue the batch job receives a run time schedule and a priority. To submit a job to a batch job queue, you use the character-based interface and one of two commands.

## **Job scheduler considerations**

When choosing a job scheduler product, you need to consider a variety of different features. The following is a list of features to consider when determining which job scheduler to use:

### • **Automated job scheduling**

- Flexibility in scheduling jobs
- Unattended (or attended) job processing 24 hours a day, 7 days a week, with total compliance to the schedules you set
- Natural extension of the IBM i operating system
- Complete control of how, when, and where a job is submitted
- Extensive job dependencies such as objects (existence of a file or records within a physical file), the activity or inactivity of other jobs, or the status of a line, controller, or subsystem
- Complete calendaring functions, including fiscal and holiday calendars
- Multiple runs per day

### • **System and user-defined parameters**

- Current® date, submission date, previous date, and current time can be passed into application programs
- User-defined parameter values can be created, changed, and passed into application programs

### • **Workload/history forecasting**

- Forecasts all scheduled jobs to be run next week, next month, or next day
- Optimize production requirements
- Historical tracking and logging of all Advanced Job Scheduler activity

### • **Network management**

- Jobs can be set up on any IBM i product in the network to run on any other IBM i product on the network
- Provides complete job history of the job on the submitting system
- Group and dependent jobs can be submitted through the network

### • **Report distribution and management**

- Routing, monitoring, and controlling of all output reports generated by Advanced Job Scheduler or IBM i operating system
- Spooled file distribution to multiple output queues or to remote systems with optional banner pages
- Spooled output can be duplicated or sent to any user on the IBM i network

### • **Security**

- Existing IBM i security can be utilized within Advanced Job Scheduler
- Specify who in your organization has authority to set up or change information about scheduled jobs
- Authority can be specified for either the individual functions of Advanced Job Scheduler or for specific jobs

- **Graphical user interface**

- Point and click capabilities when scheduling a job
- Manage jobs
- Maintain dependencies
- Track scheduler activity and log information

- **Other key features**

- Multiple commands per job
- Definition for job LDA (Local Data Area)
- Console monitor to run jobs in restricted state
- Check maximum run time for each job
- Interface directly to a message-based third-party paging system
- Provisions for full online documentation of each job
- Extensive cursor-sensitive help text on all displays

## **Job scheduling and system availability**

If the system is powered down or in restricted state when scheduled times are reached, jobs cannot be submitted from job schedule entries and the status of scheduled jobs cannot be changed. However, you can control how the system handles this situation after the system IPL or after it comes out of restricted state.

The job schedule entries and the scheduled jobs are processed in the order that the missed occurrences would have been handled normally. Work from other sources may enter the system while missed job schedule entries and scheduled jobs are being processed.

- **Job Schedule Entries:** You can control how each entry is handled by the value you specify for the recovery action of the entry. You can specify that a job still be submitted using the entry, that a job be submitted and held on the job queue, or that a job should not be submitted. If you request that a job be submitted, only one job is submitted from each entry, no matter how many submissions were missed while the system was not available.
- **Scheduled Job:** The system checks to determine if any scheduled times have passed while the system was not available. If a scheduled job with a passed time is found, the job's status is updated.

## **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 can process jobs from multiple job queues, only one subsystem can process jobs from a particular job queue at a time.

Subsystems select jobs from job queues in priority order, within limits that can 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 can create additional job queues that you need.

Job queues can be created in the system disk pool or in an independent disk pool. Jobs placed on a job queue in an independent disk pool are ended when the independent disk pool is varied off or when an IPL of the system occurs. Jobs on a job queue do not switch to a new system with an independent disk pool.

**Note:** APIs, such as Open List of Job Queues (QSP0LJBQ) and Retrieve Job Queue Information (QSPRJQBQ), can be called to get information about job queues.

### **Related concepts**

#### Managing job queues

As you manage the work on your system, you might find it necessary to manipulate jobs that are waiting in a job queue. Perhaps someone needs a job run immediately and the job is sitting in a queue at a low priority. Or maybe you need to perform some maintenance on a subsystem and want to move all of the jobs to a queue that is not associated with that particular subsystem.

### **Related tasks**

#### Clearing a job queue

When you clear a job queue, every job on the queue is deleted. This includes any jobs that are in the hold state. You can use IBM Navigator for i or the character-based interface to clear a job queue. Jobs that are running are not affected because they are considered active jobs and are no longer on the queue.

#### Creating job queues

To create a job queue, use the character-based interface.

#### Deleting a job queue

To delete a job queue, use the character-based interface.

#### Holding a job queue

When you place a job queue on hold you prevent the processing of all of the jobs that are currently waiting on the job queue. Placing a job queue on hold has no effect on jobs that are running. Additional jobs can be placed on the job queue while it is held, but they are not processed.

#### Releasing a job queue

When you release a job queue, all of the jobs that were placed on hold as a result of placing the job queue on hold are also released. If an individual job was placed on hold before the job queue was held, then the job is not released.

### **Related information**

#### Work management APIs

## **Ordered list**

The ordered list refers to the order in which jobs appear on the job queue. The availability, priority, and the date and time values help determine the order of jobs on the job queue.

The job number is not used to determine where the job appears in the job queue, nor does it affect when the job is run.

### **Availability**

Refers to the status of the job on the job queue. The possible values in order are waiting, scheduled, and held.

### **Priority**

Refers to the priority the job has on the job queue. The possible priority values are 0-9, with 0 being the highest priority. In cases where the jobs are scheduled jobs, the priority does not play a part in the order of the jobs on the job queue. For instance, if two jobs are scheduled to run at 12:00:00, the jobs are ordered by their position in the job table.

### **Date and time**

Refers to the date and time of the job:

- If the job is scheduled, the date and time refers to when the job is scheduled to run.
- If the job is not scheduled, the date and time refers to when the job entered the system.

**Note:** There are cases where the date and time end up being a date and time manually set to properly position a moved job to a particular job queue.

## How a job queue works

Job queues are allocated by a subsystem via the job queue entry. Jobs can be placed on a job queue even if the subsystem has not been started. When the subsystem is started, it processes the jobs on the queue.

The subsystem description specifies the maximum number of jobs (batch or interactive) that can be active at the same time. The number of jobs that can be active from any job queue is specified in the job queue entry.

Not all jobs on a job queue are necessarily available for processing when the subsystem is started. Scheduled jobs can be placed on the job queue. Jobs can be held on a queue until the system operator releases them. If the subsystem is ended before all of the jobs are processed, the jobs remain on the queue until the subsystem is started again, until moved by the system operator to another job queue, until deleted by the system operator, or until another subsystem allocates the same job queue.

More than one subsystem description can refer to the same job queue, but only one active subsystem at a time can use the job queue as a source of batch jobs. Therefore, if a subsystem ends and jobs are still on the job queue, another subsystem referring to that job queue can be started to process the jobs. If another subsystem is already started and is waiting for the same job queue, the subsystem automatically allocates the job queue when it becomes available.

### Related concepts

[How a subsystem handles several job queues](#)

To illustrate how a subsystem handles several job queues, consider this scenario.

### Related tasks

[Determining which subsystem has a job queue allocated](#)

You can determine which subsystem has allocated the job queue using the IBM Navigator for i interface or the character-based interface. This is useful when you find it necessary to delete the job queue since you cannot delete a job queue to which a subsystem is active.

[Creating job queues](#)

To create a job queue, use the character-based interface.

[Assigning the job queue to the subsystem](#)

To assign a job queue entry to a subsystem description, use the character-based interface.

## How jobs are taken from a job queue

Different factors determine how the jobs are selected from a job queue and started.

### Maximum active jobs for subsystems

This represents the maximum number of jobs that can be running in a subsystem. After 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. After 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.)

### 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.

## Related tasks

### Placing a job on the job queue

Jobs are placed on the job queue by either moving an existing job from one queue to another, or by submitting a new job. Use IBM Navigator for i to move jobs between queues. Use the character-based interface to submit a new job.

### Moving a job to a different job queue

There are many reasons why you might want to move a job to another queue. For example, sometimes jobs become backlogged in the queue because of a long running job. Perhaps the job's scheduled run time conflicts with a new job that has a higher priority. One way to manage this situation is to move the waiting jobs to another queue that is not as busy.

### Changing the number of jobs running simultaneously in a job queue

The QBASE subsystem is shipped with a job queue entry for the QBATCH job queue. This entry only allows one batch job to run at a time. If you want more than one batch job from that job queue to run simultaneously then you need to change the job queue entry.

## Job queue entry

A job queue entry identifies a job queue from which jobs are selected for running in the subsystem. There are five parameters in the job queue entry that control how the job queue should be handled.

### **Subsystem Description (SBSD)**

This is the name and the library of the subsystem description to which the job queue entry is added.

### **Job queue (JOBQ)**

Specifies the name and library of the job queue that is a source of batch jobs that are started by the subsystem.

### **Maximum active jobs (MAXACT)**

Specifies the maximum number of jobs that can be active at the same time from this job queue.

### **Sequence number (SEQNBR)**

Specifies a sequence number for this job queue, which is used by the subsystem to determine the order in which the job queues are processed.

### **Max active priority 1 (through 9) (MAXPTYx)**

Specifies the number of jobs that can be started for a specified job priority level.

## Related tasks

### Adding job queue entries

A job queue entry identifies a job queue from which jobs are selected for running in the subsystem. Jobs started from a job queue are batch jobs. You add a job queue entry using the character-based interface.

### Changing job queue entries

You can change an existing job queue entry in the specified subsystem description. This command can be issued while a subsystem is active or inactive. To change the job queue entry in a subsystem, use the character-based interface.

### Removing job queue entries

You can remove job queue entries from a subsystem description by using the character-based interface. Jobs on the job queue remain on the queue when the job queue entry is removed from the subsystem description. A job queue entry cannot be removed if any currently active jobs were started from the job queue.

### Changing the number of jobs running simultaneously in a job queue

The QBASE subsystem is shipped with a job queue entry for the QBATCH job queue. This entry only allows one batch job to run at a time. If you want more than one batch job from that job queue to run simultaneously then you need to change the job queue entry.

## How job queues are allocated to a subsystem

A job queue can be associated with several subsystems but it can only be allocated to one subsystem at a time. When the subsystem is started, the subsystem monitor tries to allocate each job queue defined in the subsystem job queue entries.

If a job queue was already allocated by another subsystem, the first subsystem must end and deallocate the job queue before the second subsystem can allocate it. After it is started, this second subsystem allocates job queues assigned to it as they become available.

If a job queue does not exist when the subsystem is started, the job queue is allocated to the subsystem when one of the following occurs:

- The job queue is created.
- A job queue is renamed with the name defined to the subsystem.
- A job queue is moved to another library and the resulting qualified name matches the name in the subsystem description.
- The library containing the job queue is renamed and the resulting qualified name matches the name in the subsystem description.

## Multiple job queues

In many cases, using QBATCH as the only job queue with the default of one active job will be adequate for your needs. If this is not adequate, you might want to have multiple job queues so that some job queues are active during normal working hours, some are for special purposes, and some are active after normal working hours.

For example, you can designate different job queues for:

### Long-running jobs so you can control how many jobs are active at the same time

You might also want these jobs to use a lower priority than the other batch jobs.

### Overnight jobs that are inconvenient to run during normal working hours

For example, to run a Reorganize Physical File Member (RGZPFM) command on a large database file requires an exclusive lock on the file. This means that other users cannot access the file while this operation is taking place. Additionally, this operation can take a long time. It can be more efficient to place this job on a job queue for jobs which run during off-shift hours.

### High-priority jobs

You might want to have a job queue to which all high-priority work is sent. You can then ensure that this work is completed rapidly and is not delayed by lower-priority jobs.

### Jobs that are directed to particular resource requirement such as diskette or tape

Such a job queue needs a MAXACT parameter of 1 in the job queue entry of the subsystem description so that only one job at a time uses the resource.

For example, if a tape is used for several jobs, all jobs using tape are placed on a single job queue. One job at a time are then selected from the job queue. This ensures that no two jobs compete for the same device at the same time. If this happens, one of the jobs ends with an allocation error.

**Note:** Tape output cannot be spooled.

### Programmer work

You might want to have a job queue to handle programmer work or types of work that can be held while production work is being run.

## Sequential running of a series of jobs

You can have an application in which one job is dependent on the completion of another job. If you place these jobs on a job queue that selects and runs one job at a time, this ensures the running sequence of these jobs.

If a job requires exclusive control of a file, you might want to place it on a job queue when the queue is the only one active on the server, such as during the night or on a weekend.

If you use multiple job queues, you will find that control of the various job queues is a main consideration. You will typically want to control:

- How many job queues exist
- How many job queues are active in a particular subsystem at the same time
- How many active jobs can be selected from a particular job queue at a particular time
- How many jobs can be active in a subsystem at a particular time

## How jobs are taken from multiple job queues

A subsystem processes jobs from a job queue based on sequence number. A subsystem can have more than one job queue entry and can therefore allocate more than one job queue.

The maximum number of jobs from a queue is specified by the Maximum active jobs MAXACT parameter on the Add Job Queue Entry (ADDJOBQE) or the Change Job Queue Entry (CHGJOBQE) commands. You can also control how many jobs of each priority can be active by using the Maximum active priority MAXACTx parameters. For example, if MAXACT=10, MAXACT5=2, and there are three jobs on the job queue at priority level 5, then only two of them can become active at any given time.

The subsystem processes jobs from the job queue with the lowest sequence number first. When all of the jobs that are on the job queue have been processed, or when the maximum number of jobs from the queue is reached, the subsystem processes jobs from the queue with the next higher sequence number.

The sequence continues until the subsystem has processed all of the available job queue entries or until the subsystem has reached its limit of jobs that can be running or waiting in the subsystem. The number of jobs that can be running or waiting is determined by the Maximum active jobs (MAXACT) parameter in the subsystem description. In some cases the sequence is interrupted as jobs end or are transferred. Creating, holding, and releasing job queues can also change the sequence of job queues processed.

### Related tasks

#### Placing a job on the job queue

Jobs are placed on the job queue by either moving an existing job from one queue to another, or by submitting a new job. Use IBM Navigator for i to move jobs between queues. Use the character-based interface to submit a new job.

#### Moving a job to a different job queue

There are many reasons why you might want to move a job to another queue. For example, sometimes jobs become backlogged in the queue because of a long running job. Perhaps the job's scheduled run time conflicts with a new job that has a higher priority. One way to manage this situation is to move the waiting jobs to another queue that is not as busy.

#### Changing the number of jobs running simultaneously in a job queue

The QBASE subsystem is shipped with a job queue entry for the QBATCH job queue. This entry only allows one batch job to run at a time. If you want more than one batch job from that job queue to run simultaneously then you need to change the job queue entry.

## Job queue security

You can maintain a level of security with your job queue by authorizing only certain persons (user profiles) to that job queue. In general, there are three ways that a user can become authorized to control a job queue (for example, hold or release the job queue).

- User is assigned spool control authority (SPCAUT(\*SPLCTL)) in the user profile.



- User is assigned job control authority (SPCAUT(\*JOBCTL)) in the user profile and the job queue can be controlled by the operator (OPRCTL(\*YES)).
- User has the required object authority to the job queue. The required object authority is specified by the AUTCHK parameter on the CRTJOBQ command. A value of \*OWNER indicates that only the owner of the job queue is authorized via the object authority for the job queue. A value of \*DTAAUT indicates that users with \*CHANGE authority for the job queue are authorized to control the job queue.

**Note:** The specific authority required for \*DTAAUT are \*READ, \*ADD, and \*DLT data authority.

These three methods of authorization apply only to the job queue, not to the jobs on the job queue. The normal authority rules for controlling jobs apply whether the job is on a job queue or whether it is currently running.

## 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 typically \*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 Printer device description (QPRTDEV) system value.

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 need 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 about how printer output is created, see Chapter 1 of the Printer Device Programming manual.

**Printer output files** are files that hold information waiting to be printed or processed. The printer output file 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 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 of the Output properties window.

### 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 is not closed.

#### File end (default)

Printing begins as soon as the printer output file is closed.

## Job end

Printing begins when the job ends.

After the printer output file 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 and sends it to the designated printer.

## Related concepts

[Managing 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.

## Related information

[Experience Report: Spool Performance Considerations](#)

[Basic Printing](#)

## 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.

Because most of the information that you print on your 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 to an output queue or printer output file. 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.

### 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 that 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 about IBM i authorities, see [Authority required for objects used by commands in the Security reference topic collection](#).

## Order of files

The Order of files on the queue (SEQ) attribute determines how the printer output will leave the output queue to be processed.

This attribute has two values:

- \*FIFO: The queue is first-in first-out within priority for each file. That is, new spooled files are placed after all other entries on the queue of the same priority.

- **\*JOBNBR** : The queue entries for spooled files are sorted in priority sequence using the job number (actually, the date and time that the job entered the system is used) of the job that created the spooled file.

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

## Spooled files

Spooling is a system function that saves data for later processing or printing. This data is stored in a spooled file. Spooled files work in a similar manner to tape files or other device files. Spooled files allow you to manage your data targeted for externally attached devices such as a printer.

Spooling functions help server users to manage input and output operations more efficiently. The server supports two types of spooling, output spooling and input spooling. Output spooling can be used for printer devices. Input spooling applies to database file input.

### Related information

[Spooled files and output queues](#)

### **Output spooling**

Output spooling can be used for both printer and diskette devices. Output spooling sends job output to disk storage instead of sending it directly to a printer or diskette output device. Output spooling allows the job that produces the output to continue processing without consideration for the speed or availability of output devices.

Additionally, output spooling allows the server to produce output on multiple output devices, such as printer and diskette devices, in an efficient manner. It does this by sending the output of a job destined for a printer to disk storage. This process breaks a potential job limitation imposed by the availability or speed of the output devices.

The main elements of output spooling are:

- **Device description:** A description of the printer device.
- **Spooled file:** A file containing spooled output records that are to be processed on an output device.
- **Output queue:** An ordered list of spooled files.
- **Writer:** A program that sends files from an output queue to a device.
- **Application program:** A high-level language program that creates a spooled file using a device file with the spooling attribute specified as SPOOL(\*YES).
- **Device file:** A description of the format of the output and a list of attributes that describe how the server should process the spooled file.

Output spooling functions are performed by the server without requiring any special operations by the program that produces the output. When a device file is opened by a program, the operating system determines whether the output is to be spooled. When a printer file that specifies spooling is opened, the spooled file that contains the output of the program is placed on the appropriate output queue in the server.

A spooled file can be made available for printing when the printer file is opened, when the printer file is closed, or at the end of the job. A printer writer is started in the spooling subsystem to send the records to the printer. The spooled file is selected from an output queue.

## Spooling device descriptions

Device descriptions must be created for each printer and diskette device in order to define that device to the server. Printer device descriptions are created using the Create Device Description for Printer (CRTDEVPRT) command; diskette device descriptions are created using the Create Device Description for Diskette (CRTDEVDKT) command.

## File redirection of spooled files

File redirection occurs when a spooled file is sent to an output device other than the one for which it was originally intended. File redirection can involve devices that process different media (such as printer output sent to a diskette device) or devices that process the same type of media but are of different device types (such as 5219 Printer output sent to a 4224 Printer).

Depending on the new output device for the spooled file, the file can be processed just as it would have been on the originally specified device. However, differences in devices often cause the output to be formatted differently. In these cases, the server sends an inquiry message to the writer's message queue to inform you of the situation and allow you to specify whether you want printing to continue.

### ***Output queues and spooled files***

Batch and interactive job processing can result in spooled output records that are to be processed on an output device, such as a printer or diskette drive. These output records are stored in spooled files until they can be processed. A single job can have many spooled files.

When a spooled file is created, the file is placed on an output queue. Each output queue contains an ordered list of spooled files. A job can have spooled files on one or more output queues. All spooled files on a particular output queue should have a common set of output attributes, such as device, form type, and lines per inch. Using common attributes on an output queue reduces the amount of intervention required and increases the device throughput.

The following lists some of the parameters on the Create Output Queue (CRTOUTQ) command and what they specify:

- **MAXPAGES:** Specifies the maximum spooled file size in pages that is allowed to be printed between a starting and ending time of day.
- **AUTOSTRWTR:** Specifies the number of writers that are started automatically to this output queue.
- **DSPDTA:** Whether users without any special authority but who do have \*USE authority to the output queue can display, copy, or send the contents of spooled files other than their own. By specifying \*OWNER for DSPDTA, only the owner of the file or a user with \*SPLCTL special authority can display, copy, or send a file.
- **JOBSEP:** The number of job separator pages, if any, that are to be printed between the output of each job when the output is printed.
- **DTAQ:** The data queue associated with this output queue. If specified, an entry is sent to the data queue whenever a spooled file goes to ready status on the queue.
- **OPRCTL:** Whether a user who has job control authority can control the output queue (for example, if the user can hold the output queue).
- **SEQ:** Controls the order in which spooled files are sorted on the output queue.
- **AUTCHK:** Specifies what type of authority to the output queue that enables a user to control the spooled files on the output queue (for example, enables a user to hold the spooled files on the output queue).
- **AUT:** Public authority. Specifies what control users have over the output queue itself.
- **TEXT:** Text description. Up to 50 characters of text that describes the output queue.

### ***Default system output queues***

Defaults on CL commands use the default output queue for the system printer as the default output queue for all spooled output. The system printer is defined by the QPRTDEV server value.

When a spooled file is created by opening a device file and the output queue specified for the file cannot be found, the system attempts to place the spooled file on output queue QPRINT in library QGPL. If for any reason the spooled file cannot be placed on output queue QPRINT, an error message is sent and the output is not spooled.

The following output queues are provided:

- **QDKT:** Default diskette output queue
- **QPRINT:** Default printer output queue

- **QPRINTS**: Printer output queue for special forms
- **QPRINT2**: Printer output queue for 2-part paper

### ***Spooling writers***

A writer is an IBM i program that takes spooled files from an output queue and produces them on an output device. The spooled files that have been placed on a particular output queue remain stored in the system until a writer is started to the output queue.

The writer takes spooled files one at a time from the output queue, based on their priority. The writer processes a spooled file only if its entry on the output queue indicates that it has a ready (RDY) status. You can display the status of a particular spooled file using the Work with Output Queue (WRKOUTQ) command.

If the spooled file has a ready status, the writer takes the entry from the output queue and prints the specified job or file separators or both, followed by the output data in the file. If the spooled file does not have a ready status, the writer leaves the entry on the output queue and goes on to the next entry. In most cases the writer continues to process spooled files (preceded by job and file separators) until all files with a ready status have been taken from the output queue.

The AUTOEND parameter on the start writer commands determines whether the writer continues to wait for new spooled files to become available to be written, end after processing one file, or end after all spooled files that have a ready status have been taken from the output queue.

### ***Spooling writer commands***

Here are the commands that you can use to control spooling writers.

- Start Diskette Writer (STRDKTWTR): Starts a spooling writer to a specified diskette device to process spooled files on that device.
- Start Printer Writer (STRPRTWTR): Starts a spooling writer to a specified printer device to process spooled files on that device.
- Start Remote Writer (STRRMTWTR): Starts a spooling writer that sends spooled files from an output queue to a remote system.
- Change Writer (CHGWTR): Changes some writer attributes, such as form type, number of file separator pages, or output queue attributes.
- Hold Writer (HLDWTR): Stops a writer at the end of a record, at the end of a spooled file, or at the end of a page.
- Release Writer (RLSWTR): Releases a previously held writer for additional processing.
- End Writer (ENDWTR): Ends a spooling writer and makes the associated output device available to the server.

**Note:** You can define some functions to provide additional spooling support. Example source and documentation for the commands, files, and programs for these functions are part of library QUSRTOOL, which is an optionally installed part of IBM i.

### ***Related information***

[Start Printer Writer \(STRPRTWTR\) command](#)  
[Start Remote Writer \(STRRMTWTR\) command](#)  
[Change Writer \(CHGWTR\) command](#)  
[Hold Writer \(HLDWTR\) command](#)  
[Release Writer \(RLSWTR\) command](#)  
[End Writer \(ENDWTR\) command](#)

### ***Input spooling***

Input spooling takes the information from the input device, prepares the job for scheduling, and places an entry in a job queue. Using input spooling, you can typically shorten job run time, increase the number of jobs that can be run sequentially, and improve device throughput.

The main elements of input spooling follow:

- **Job queue:** An ordered list of batch jobs submitted to the system for running and from which batch jobs are selected to run.
- **Reader:** A function that takes jobs from an input device or database file and places them on a job queue.

When a batch job is read from an input source by a reader, the commands in the input stream are stored in the system as requests for the job, the inline data is spooled as inline data files, and an entry for the job is placed on a job queue. The job information remains stored in the system where it was placed by the reader until the job entry is selected from the job queue for processing by a subsystem.

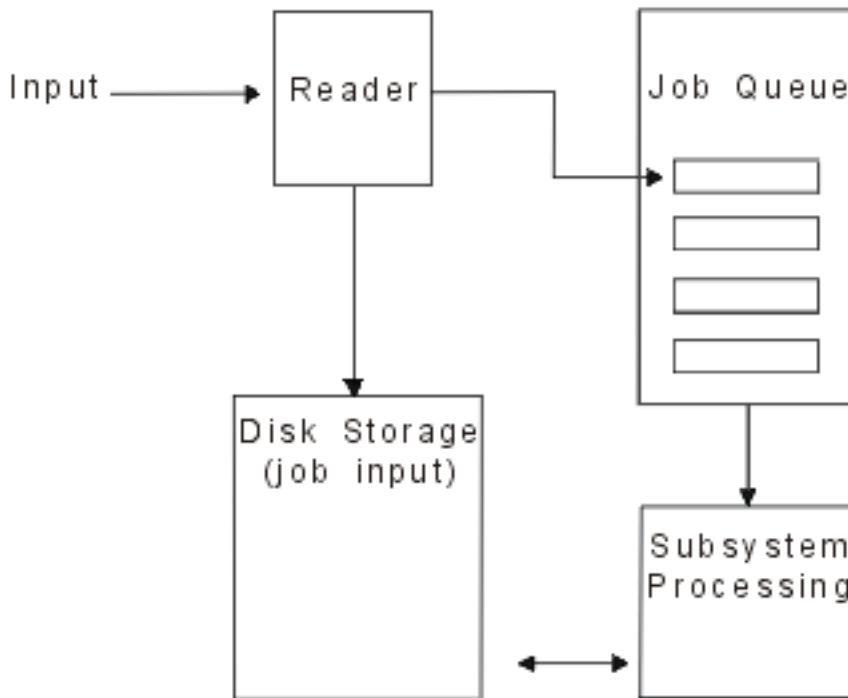


Figure 2. Relationship of input spooling elements

You can use the reader functions to read an input stream from diskette or database files.

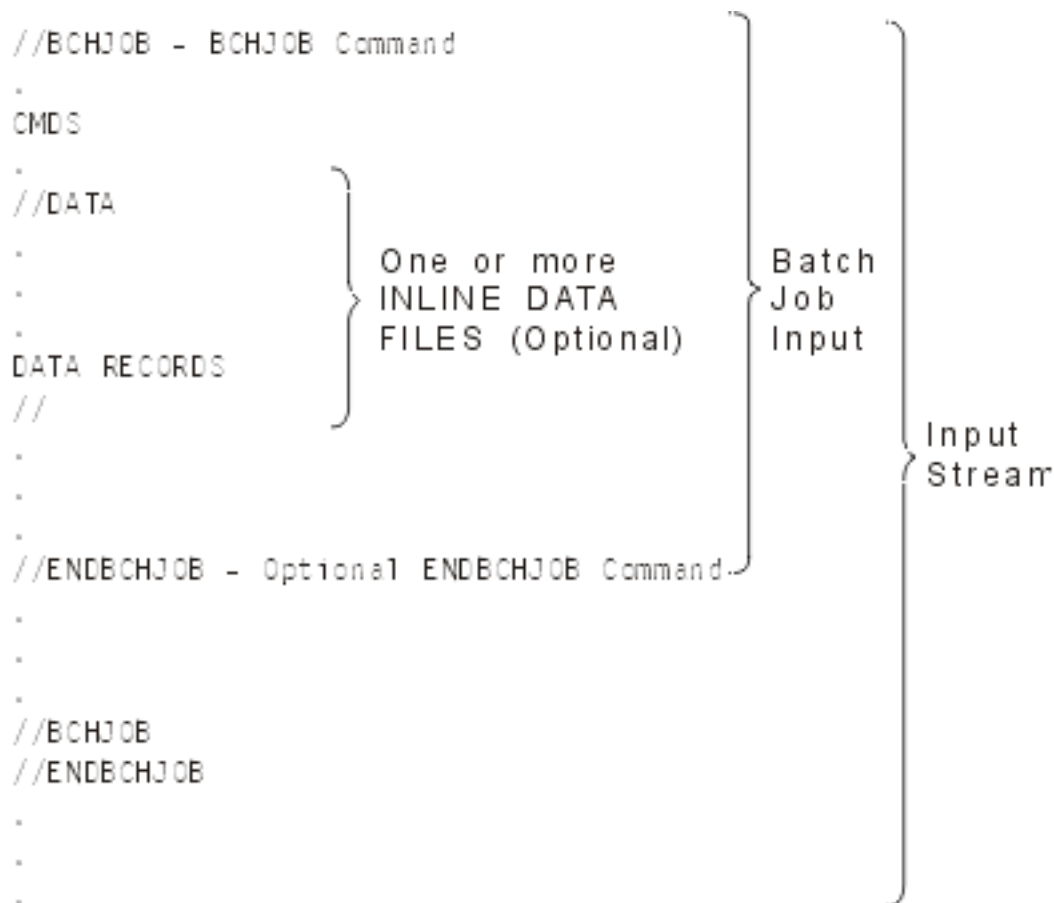


Figure 3. Typical organization of an input stream

The job queue on which the job is placed is specified on the JOBQ parameter of the Batch Job BCHJOB or the Start Database Reader STRDBRDR command, or in the job description. The values of the JOBQ parameter for the BCHJOB command follow:

- \*RDR: The job queue is selected from the JOBQ parameter on the STRDBRDR command.
- \*JOBQ: The job queue is selected from the JOBQ parameter in the job description.
- A specific job queue: The specified queue is used.

For jobs with small input streams, you might improve system performance by not using input spooling. The Submit Job (SBMJOB) command reads the input stream and places the job on the job queue in the appropriate subsystem, bypassing the spooling subsystem and reader operations.

If your job requires a large input stream to be read, you should use input spooling (Start Diskette Reader STRDKTRDR or STRDBRDR command) so that the job can be imported independent of when the job is actually processed.

### Job input commands

You can use these commands to submit jobs to the system. The start reader commands can be used for spooling job input; the submit job commands do not use spooling.

- Batch Job (BCHJOB): Marks the start of a job in a batch input stream and defines the operating characteristics of the job.
- Data (DATA): Marks the start of an inline data file.
- End Batch Job (ENDBCHJOB): Marks the end of a job in a batch input stream.
- End Input (ENDINP): Marks the end of the batch input stream.
- Submit Database Jobs (SBMDBJOB): Reads an input stream from a database file and places the jobs in the input stream on the appropriate job queues.

- Submit Diskette Jobs (SBMDKTJOB): Reads an input stream from diskette and places the jobs in the input stream on the appropriate job queues.
- Start Database Reader (STRDBRDR): Starts a reader to read an input stream from a database file and places the job in the input stream on the appropriate job queue.
- Start Diskette Reader (STRDKTRDR): Starts a reader to read an input stream from diskette and places the job in the input stream on the appropriate job queue.

### Related information

[CL command finder](#)

[Batch Job \(BCHJOB\) command](#)

[Data \(DATA\) command](#)

[End Batch Job \(ENDBCHJOB\) command](#)

[End Input \(ENDINP\) command](#)

[Submit Data Base Jobs \(SBMDBJOB\) command](#)

[Start Data Base Reader \(STRDBRDR\) command](#)

### Inline data files

An inline data file is a data file that is included as part of a batch job when the job is read by a reader or a submit jobs command. You use SBMDBJOB or STRDBRDR to queue a CL batch stream (stream of CL commands to be run). That CL batch stream can include data to be placed into inline data files (temporary files). When the job ends, the inline data files are deleted.

An inline data file is delimited in the job by a //DATA command at the start of the file and by an end-of-data delimiter at the end of the file.

The end-of-data delimiter can be a user-defined character string or the default of //. The // must appear in positions 1 and 2. If your data contains // in positions 1 and 2, you should use a unique set of characters, such as // \*\*\* END OF DATA. To specify this as a unique end-of-data delimiter, the ENDCHAR parameter on the //DATA command should be coded as:

```
ENDCHAR('// *** END OF DATA')
```

**Note:** Inline data files can be accessed only during the first routing step of a batch job. If a batch job contains a Transfer Job (TFRJOB), a Reroute Job (RRTJOB), or a Transfer Batch Job (TFRBCHJOB) command, the inline data files cannot be accessed in the new routing step.

An inline data file can be either named or unnamed. For an unnamed inline data file, either QINLINE is specified as the file name in the //DATA command or no name is specified. For a named inline data file, a file name is specified.

A *named inline data file* has the following characteristics:

- It has a unique name in a job. No other inline data file can have the same name.
- It can be used more than once in a job.
- Each time it is opened, it is positioned to the first record.

To use a named inline data file, you must either specify the file name in the program or use an override command to change the file name specified in the program to the name of the inline data file. The file must be opened for input only.

An *unnamed inline data file* has the following characteristics:

- Its name is QINLINE. (In a batch job, all unnamed inline data files are given the same name.)
- It can only be used once in a job.
- When more than one unnamed inline data file is included in a job, the files must be in the input stream in the same order as when the files are opened.

To use an unnamed inline data file, do one of the following:

- Specify QINLINE in the program.



- Use an override file command to change the file name that is specified in the program to QINLINE.

If your high-level language requires unique file names within one program, you can use QINLINE as a file name only once. If you need to use more than one unnamed inline data file, you can use an override file command in the program to specify QINLINE for additional unnamed inline data files.

**Note:** If you run commands conditionally and process more than one unnamed inline data file, the results cannot be predicted if the wrong unnamed inline data file is used.

### **Related concepts**

Considerations for opening inline data files

You need to consider these elements when you open inline data files.

### ***Considerations for opening inline data files***

You need to consider these elements when you open inline data files.

- The record length specifies the length of the input records. (The record length is optional.) When the record length exceeds the length of the data, a message is sent to your program. The data is padded with blanks. When the record length is less than the data length, the records are truncated.
- When a file is specified in a program, the system searches for the file as a named inline data file before it searches for the file in a library. Therefore, if a named inline data file has the same name as a file that is not an inline data file, the inline data file is always used, even if the file name is qualified by a library name.
- Named inline data files can be shared between programs in the same job by specifying SHARE(\*YES) on a create file or override file command. For example, if an override file command specifying a file named INPUT and SHARE(\*YES) is in a batch job with an inline data file named INPUT, any programs running in the job that specify the file name INPUT shares the same named inline data file. Unnamed inline data files cannot be shared between programs in the same job.
- When you use inline data files, make sure the correct file type is specified on the //DATA command. For example, if the file is to be used as a source file, the file type on the //DATA command must be source.
- Inline data files must be opened for input only.

### **Related concepts**

Inline data files

An inline data file is a data file that is included as part of a batch job when the job is read by a reader or a submit jobs command. You use SBMDBJOB or STRDBRDR to queue a CL batch stream (stream of CL commands to be run). That CL batch stream can include data to be placed into inline data files (temporary files). When the job ends, the inline data files are deleted.

## **Job logs**

A job log contains information related to requests entered for a job. A job log has two forms, a pending form and a spooled form.

In its pending form, a job log for a completed job can change as other jobs (the subsystem, the system operator, and so on) interact with the completed job. In its spooled form, a job log is a snapshot (a moment in time) and does not change (such as spooled files that are created by the Display Job Log (DSPJOBLOG) command, or created after the job completes its activity).

Each job has an associated job log that can contain the following information for the job:

- The commands in the job
- The commands in a CL program (if the CL program was created with the LOG(\*YES) option or with the LOG(\*JOB) option and a Change Job (CHGJOB) command was run with the LOGCLPGM(\*YES) option)
- All messages (the message and help text for the message) sent to the requester and not removed from the program message queues

At the end of the job, the job log can be written to the spooled file QPJOBLOG so that it can be printed. However, producing a job log doesn't necessarily mean printing it or creating a spooled file. (For example,

the Control Job Log QMHCTLJL API can be used to specify that the job log is to be written to an outfile at the end of job.)

You can reduce the number of job logs produced and reduce the contention for resources (such as output queues). This reduces the resource consumption caused by producing job logs.

### **Related concepts**

#### Managing job logs

Most jobs on your system have a job log associated with it. Job logs tell you 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 you a good idea of how the job cycle is running.

#### Managing the job log server

The QSYSWRK subsystem controls the job log server. However, there are some tasks that you can perform to customize or manage the job log server.

### **Related tasks**

#### Deleting job log output files

Job logs are removed from the system when a job completes normally, or when the Remove Pending Job Log (QWTRMVJL) API or the End Job (ENDJOB) command is issued. Additionally if "clear incomplete job logs" is specified on the IPL, all of the jobs in job log pending are removed from the system during an IPL. Any remaining job log output files can be found under **Basic Operations > Printer Output**.

#### Controlling batch job log information

For your batch applications, you may want to change the amount of information logged. The log level (LOG(40 \*NOLIST)) specified in the job description for the IBM-supplied subsystem QBATCH supplies a complete log if the job abnormally ends. If the job completes normally, no job log is produced.

#### Changing the log level of a job

The log level of a job is a numeric level assigned to a specific combination of message types that are logged. You can change the log level in the job description by using the character-based interface. However, if you want to change the log level of a specific job, use the **Job Properties - Job Log** window in IBM Navigator for i.

### **Related information**

[Experience Report: Spool Performance Considerations](#)

## **How job logs are created**

The job logs are available when needed, but no work is done to produce job logs for which there is no need.

The LOG parameter has three elements: the message (or logging) level, the message severity, and the level of message text. Each of these elements have specific values that when combined determine the amount and type of information sent to the job log by the job.

For example, the \*NOLIST value of the Text element causes no job log to be produced if the job ends normally. (The job log does not go into pending.) If the job ends abnormally (if the job end code is 20 or higher), a job log is produced. The messages that appear in the job log contain both the message text and the message help.

You can control what produces the job log. This is done with the LOGOUTPUT parameter. When a job completes, one of three actions occur that affects how the job log is created. The following are values of the LOGOUTPUT parameter:

- **The job log server produces the job log:** (\*JOBLOGSVR)
- **The job itself produces the job log:** If the job cannot produce its own job log, the job log is produced by a job log server. (\*JOBEND)
- **The job log is not produced:** The job log remains in pending until it is removed. (\*PND)

**Note:** These values do not affect job logs that are produced when the message queue is full and the job message queue full action specifies \*PRTWRAP. Messages in the job message queue are written to a

spooled file, from which the job log can be printed, unless the Control Job Log Output (QMHCTLJL) API was used in the job to specify that the messages in the job log are to be written to a database file.

## What controls the job log parameters?

When a job starts, it gets its LOGOUTPUT value from the job description. If the job description specifies \*SYSVAL (the default for CRTJOB), the job uses the job log output value that is specified in the Job log output (QLOGOUTPUT) system value. (While the shipped value for the Job log output (QLOGOUTPUT) system value is \*JOBEND, the recommended value is \*JOBLOGSVR.) After the job has established its LOGOUTPUT job attribute, any changes to the job description or system value do not affect the active job. Changes to the system value or to the job description take effect for jobs entering the system after the change.

You can use the Change Job (CHGJOB) command or API (QWTCHGJB) to change the LOGOUTPUT job attribute after it has already been set in the job. Changes to the job take effect immediately.

Regardless of the method that you choose, the options for handling job logs are the same. You can set the job to not produce a job log (\*PND), have the job produce the job log (\*JOBEND), or have the job log server produce the job log (\*JOBLOGSVR).

### Related tasks

#### Stopping production of a specific job log

If you only want to stop the production of a particular job log, do not use the End Job Log Server (ENDLOGSVR) command. The ENDLOGSVR command ends all job log servers which results in stopping the production of all job logs.

#### Preventing the production of a job log

Preventing the production of a job log is useful if you already know that you will not need the job log and you want to conserve system resources. When you specify that you do not want to produce a job log, the job log will not be produced and remains in pending until removed either by the Remove Pending Job Log (QWTRMVJL) command or the End Job (ENDJOB) command.

#### Controlling information in a job log

When working with problems, you might want to record the maximum amount of information for jobs that have frequent problems. Alternatively, you might not want to create a job log for jobs that completed normally. Or you might want to exclude informational messages.

## Job log pending

The job log pending state has been available for many years. When the job log attribute of a job is \*PND, no job log is produced. You can control how and under what circumstances the job log for a specific job is produced.

This feature is useful when you place the system into a restricted state. When the system goes into a restricted state, subsystems end and a potential of thousands of jobs can end at once. This in turn can create a large burden on the output resources. By preventing the production of these job logs, you can significantly reduce the impact on these resources.

Another example of when you can use this feature is during a communications failure. Perhaps there are many similar jobs that produce the same job log error messages. You can set the job log to not produce a spooled file for all of the jobs. Then if there happens to be a communications failure, you can use the Work with Job Log (WRKJOBLOG) command to determine which logs to print. You can also use the Work with Job Logs (WRKJOBLOG) screen to manage job logs.

Jobs might be in a job log pending state due to the workings of the Power Down System (PWRDWN SYS) command. The IBM Navigator for i user interface shows the status "Completed - Job log pending" for these jobs. This is a subset of character-based interface status of \*OUTQ.

Taking advantage of these enhancements can help you to reduce the number of job logs produced and thereby reduce the contention for resources. This can result in improved system performance.

### Related concepts

When jobs end at the same time

Sometimes, jobs end at the same time. For example, a network error occurs and the job attributes are set to \*ENDJOB or \*ENDJOBNO LIST. In addition to the job ending, the following device recovery actions occur.

### **Related tasks**

#### Cleaning up job log pending

There are a few ways to clean up, or remove jobs from job log pending. You can end the job with a value of 0 for the Maximum log entries (LOGLMT) parameter. If the job is already ended, you can run the Remove Pending Job Log (QWTRMVJL) API. You can also use the Work with Job Logs (WRKJOBLOG) command.

#### Producing printer output from job log pending

Jobs that do not have the IBM Navigator for i **Job Properties - Job Log** setting, **Produce a job log** field selected do not produce job logs. Instead the job log is in job log pending. To produce printer output from a job log that is in job log pending, use the character-based interface.

## **Job log server**

Typically the job log server writes a job's job log to a spooled file. You can route the job log to a printer or to an outfile, (if specified to do so by using the QMHCTLJL, Control job log API), however this is not the recommended method for producing job logs.

You can view information about the job log server via IBM Navigator for i from the **Work Management > Server Jobs** display, or the **Work Management > Active Jobs** display. (To make it easier to identify the jobs running on the job log server, make sure that you include the Server column in your display.)

The maximum number of job log servers that can be active at one time is 30. You start additional job log servers and manage them in the same way as other servers in your system. This is done by using the character-based interface command STRLOGSVR.

## **How the job log server starts**

By default, the job log server will start automatically when the QSYSWRK subsystem starts. The server ends whenever the QSYSWRK subsystem is ended.

The Start Job Log Server (STRLOGSVR) command starts the job log server. The job log server writes job logs for jobs that are in a job log pending state and that do not have the attribute of \*PND. The job log server writes a job's job log either to a spooled file, to a printer, or to an outfile, (if specified to do so by using the QMHCTLJL, Control job log API).

### **Related tasks**

#### Reconfiguring the job log server

As shipped, the job log server runs in QSYSWRK. QSYSWRK is continuously active. To enhance performance, you might want to reconfigure your job log server to run in a different subsystem.

#### Starting the job log server

By default, the job log server automatically starts when the QSYSWRK subsystem starts. You can manually start a job log server by using the Start Job Log Server (STRLOGSVR) command.

#### Ending the job log server

The End Job Log Server (ENDLOGSVR) command is used to end the job log server(s). The job log server writes job logs for jobs that are in a job log pending state. If more than one job log server job is active at the time this command is issued, all of the job log server jobs are ended.

### **Related information**

Control Job Log Output (QMHCTLJL) API

## Job log display characteristics

IBM Navigator for i provides you with a user friendly, easy to read interface from which you can view job logs and the job log's messages. You can also view job logs by using the character based interface.

You can control which columns appear in the job log list by using the Job Log - Columns window. (**Work Management > Active Jobs > Right-click a job and select Job Log > Actions menu > Columns**) The columns that you can choose to display in the job log list are:

Message ID	From Program
Message	Request Level
Sent	Severity
Thread	To Program
Type	

## Character-based interface

When you use the Display Job Log (DSPJOBLOG) command, you see the Job Log display. This display shows program names with special symbols, as follows:

>>	The running command or the next command to be run. For example, if a CL or high-level language program was called, the call to the program is shown.
>	The command has completed processing.
..	The command has not yet been processed.
?	Reply message. This symbol marks both those messages needing a reply and those that have been answered.

## Job log headings

Job log headings are located at the top of each page of the printed job log. These headings identify the job to which the job log applies and the characteristics of each entry. The following is a list of possible entries in the job log heading.

- The fully qualified name of the job (job name, user name, and the job number)
  - The name of the job description used to start the job
  - The date and time the job started
  - The message identifier
  - The message type
  - The message severity
  - The date and time each message was sent
  - The message. If the logging level specifies that second-level text is to be included, the second-level text appears on subsequent lines below the message
  - The program from which the message or request was sent
  - The machine interface instruction number or the offset to the program to which the message was sent
- Note:** The machine interface instruction numbers appear only for escape, notify, and diagnostic messages. For all other message types, the machine interface instruction number is set to zero.
- If the job uses APPC, the heading contains a line showing the unit of work identifier for APPC.

## Messages

Messages contain the job name, the message type, the date and time it was sent, the action that occurred, and the necessary actions needed to fix a problem. This is useful when you are trying to troubleshoot any problems that might occur on your servers. You can access job logs for server jobs

through IBM Navigator for i. Messages fall into two categories, alertable messages and messages logged in a job log.

**Alertable messages** - These messages are sent to QSYSOPR because they need immediate action. The message contains the problem, the cause, and the recovery action necessary. For example, the server fails to start or the server ends unexpectedly. Some servers send alertable messages to QSYSOPR. These messages have the Alert Option (ALROPT) defined in the message description. You can use alerts to provide centralized handling of alertable messages.

**Messages logged in a job log** - These messages are diagnostic in nature, meaning that they are not critical but are alerting the user of some action that was taken. These messages can be system generated as well as user created.

## Message logging level

The message logging level determines which messages and what message types should be logged for the job. The following table explains what each level represents.

Level	Description
Level 1	All messages sent to the job's external message queue with a severity greater than or equal to the message severity value. (In IBM Navigator for i, the Message severity (0-99) value can be found on the Job Properties - Job Log window. This is a value that you can control.)
Level 2	All messages that meet Level 1 qualifications and any request messages which result in a high level message greater than or equal to the message severity value.  <b>Note:</b> A high-level message is one that is sent to the program message queue of the program that receives the request message. (For example, QCMD is an IBM-supplied request processing program that receives request messages.)
Level 3	All messages that meet Level 1 or Level 2 qualifications and all request messages. Additionally, any commands from CL programs are included if the <b>Log commands from CL programs box</b> is checked (Job Properties - Job Log window).  <b>Note:</b> The <b>Log commands from CL programs box</b> is equivalent to the log attribute of the CL program.
Level 4	All request messages and all messages with a severity greater than or equal to the message logging severity, including trace messages. Additionally, any commands from CL programs are included if the <b>Log commands from CL programs box</b> is checked (Job Properties - Job log window).  <b>Note:</b> The <b>Log commands from CL programs box</b> is equivalent to the log attribute of the CL program.

### Related tasks

#### Changing the log level of a job

The log level of a job is a numeric level assigned to a specific combination of message types that are logged. You can change the log level in the job description by using the character-based interface. However, if you want to change the log level of a specific job, use the **Job Properties - Job Log** window in IBM Navigator for i.

### Interactive job logs

The IBM-supplied job descriptions QCTL, QINTER, and QPGMR all have a log level of LOG(4 0 \*NOLIST); therefore all messages help text are written to the job log. However, the job logs are not printed if the job ends normally unless you specify \*LIST on the SIGNOFF command.

If a display station user uses an IBM-supplied menu or the command entry display, all error messages are displayed. If the display station user uses a user-written initial program, any unmonitored message

causes the initial program to end and a job log to be produced. However, if the initial program monitors for messages, it receives control when a message is received. In this case, it is important to ensure that the job log is produced so that you can determine the specific error that occurred.

For example, assume that the initial program displays a menu that includes a signoff option, which defaults to \*NOLIST. The initial program monitors for all exceptions and includes a Change Variable (CHGVAR) command that changes the signoff option to \*LIST if an exception occurs:

```
PGM
DCLF MENU
DCL &SIGNOFFDPT TYPE(*CHAR) LEN(7)
VALUE(*NOLIST)
.
.
.
MONMSG MSG(CPF0000) EXEC(GOTO ERROR)
PROMPT: SNDRCVF RCFMT(PROMPT)
CHGVAR &IN41 '0'
.
.
.
IF (&OPTION *EQ '90') SIGNOFF
LOG(&SIGNOFFOPT);
.
.
.
GOTO PROMPT
ERROR: CHGVAR&SIGNOFFOPT '*LIST'
CHGVAR &IN41 '1'
GOTO PROMPT
ENDPGM
```

If an exception occurs, the CHGVAR command changes the option on the SIGNOFF command to \*LIST and sets on an indicator. This indicator can be used to condition a constant that displays a message explaining that an unexpected error occurred and telling the display station user what to do.

## QHST History Log

The history (QHST) log consists of a message queue and a physical file known as a log-version. Messages sent to the log message queue are written by the system to the current log-version physical file.

The history log (QHST) contains a high-level trace of system activities such as system, subsystem, job information, device status, and system operator messages. Its message queue is QHST.

## Log-Version

Each log-version is a physical file that is named in the following way:

```
Qxxxyyddn
```

Where:

**xxx** is a 3 character description of the log type (HST)

**yyddd** is the Julian date on which the log-version was created

**n** is a sequence number within the Julian date (0 through 9 or A through Z)

When a log-version is full, a new version of the log is automatically created.

**Note:** The number of records in the log-version of the history log is specified in the Maximum records in history log (QHSTLOGSIZ) system value. This system value also supports a \*DAILY option which creates a new version each day.

## Format of the History Log

A database file is used to store the message sent to a system log. Because all records in a physical file have the same length and messages sent to a log have different lengths, the messages can span more than one record.

Each record for a message has three fields:

- System date and time (a character field of length 8). This is an internal field. The converted date and time also are in the message.
- Record number (a 2-byte field). For example, the field contains hex 0001 for the first record, hex 002 for the second record, and so on.
- Data (a character field of length 132).

### Format for the third field (data):

Table 2. Format for Third Field of the First Record

Contents	Type	Length	Positions in Record
Job name	Character	26	11-36
Converted date and time	Character	13	37-49
Message ID	Character	7	50-56
Message file name	Character	10	57-66
Library name	Character	10	67-76
Message type	Character	2	77-78
Severity code	Character	2	79-80
Sending program name	Character	12	81-92
Receiving program name	Character	10	97-106
Receiving program instruction number	Character	4	107-110
Message text length	Binary	2	111-112
Message data length	Binary	2	113-114
Reserved	Character	28	115-142

Table 3. Format of the third field (data) of the remaining records

Contents	Type	Length
Message	Character	Variable (This length is specified in the first record (positions 111 and 112) and cannot exceed 132.)
Message data	Character	Variable (This length is specified in the first record (positions 113 and 114).)

A message is never split when a new version of a log is started. The first and last records of a message are always in the same QHST version.



## **Performance information and QHST**

Performance information is not displayed as text on message CPF1164. Because the message is in the QHST log, users can write application programs to retrieve this data.

The performance information is passed as a variable length replacement text value. This means that the data is in a structure within the first entry being the length of the data. The size of the length field is not included in the length.

**Time and Date:** The first data fields in the structure are the times and dates that the job entered the system and when the first routing step for the job was started. The times are in the format 'hh:mm:ss'. The time separators in this example are colons. This separator is determined by the value specified in the Date and time (QTIMSEP) system value. The dates are in the format defined in the Date and time (QDATFMT) system value and the separators are in the Date and time (QDATSEP) system value. The time and date the job entered the system precede the job start time and date in the structure. The time and date the job entered the system are when the system becomes aware of a job to be initiated (a job structure is set aside for the job). For an interactive job, the job entry time is the time the password is recognized by the system. For a batch job, it is the time the Batch Job (BCHJOB) or Submit Job (SBMJOB) command is processed. For a monitor job, reader or writer, it is the time the corresponding start command is processed, and for autostart jobs it is during the start of the subsystem.

**Total Response Time and Number of Transactions:** Following the times and dates are the total response time and the number of transactions. The total response time is in seconds and contains the accumulated value of all the intervals the job was processing between pressing the Enter key at the workstation and when the next display is shown. This information is similar to that shown on the Work with Active Job (WRKACTJOB) display. This field is only meaningful for interactive jobs.

It is also possible in the case of a system failure or abnormal job end that the last transaction will not be included in the total. The job end code in this case would be a 40 or greater. The transaction count is also only meaningful for interactive jobs other than the console job and is the number of response time intervals counted by the system during the job.

**Number of Synchronous Auxiliary I/O Operations:** The number of synchronous auxiliary I/O operations follows the number of transactions. This is the same as the AUXIO field that appears on the WRKACTJOB display except for the following difference:

- The WRKACTJOB display shows the value for the initial thread of the current routing step.
- The QHST message contains the cumulative total for the job of each routing step in the job.

If the job ends with an end code of 70, this value may not contain the count for the final routing step. Additionally, if a job exists across an IPL (using a Transfer Batch Job (TFRBCHJOB) command) it is ended before becoming active following an IPL, the value is 0.

## **Spooled files**

A spooled file holds output data until it can be printed. The spooled file collects data from a device until a program or device is able to process the data. A program uses a spooled file as if it were reading from or writing to an actual device. This is input and output spooling.

Input spooling is done by the system for database and diskette files. An IBM-supplied program, called a reader, is started in the spooling subsystem, reads the batch job streams from the device, and places the jobs on a job queue.

Output spooling is done for printers. An IBM-supplied program, called a printer writer, is started in the spooling subsystem, selects spooled files from its output queue, and writes the records of the spooled output file to the printer.

At the end of a the job, the job log can be written to the spooled file QPJOBLOG so that it can be printed.

## **Job accounting**

The job accounting function gathers data so that you can determine who is using your system and what system resources they are using. It also assists you in evaluating the overall use of your system.

Job accounting is optional. You must take specific steps to set up job accounting. You can request the system to gather job resource accounting data, printer file accounting data, or both. You can also assign accounting codes to user profiles or specific jobs.

Typical job accounting data details the jobs running in your system and the resources they are using such as the use of the processing unit, printer, display stations, database and communications functions.

Job accounting statistics are kept by using the journal entries made in the system accounting journal QSYS/QACGJRN. You should know how to perform journal management operations, such as saving a journal receiver, changing journal receivers, and deleting old journal receivers.

When you want to analyze the job accounting data, it must be extracted from the QACGJRN journal by use of the Display Journal (DSPJRN) command. With this command you can write the entries into a database file. You must write application programs or use a utility such as the query utility to analyze the data.

### **Related concepts**

[Managing job accounting](#)

The job accounting function is not active by default. It requires a few initial steps to set it up. The following information describes how to set up job accounting and perform some of the most common tasks associated with job accounting.

### **Related information**

[Journal Management](#)

[Set up journaling](#)

## How job accounting works

For this overview of how job accounting works, assume three different jobs enter the system.

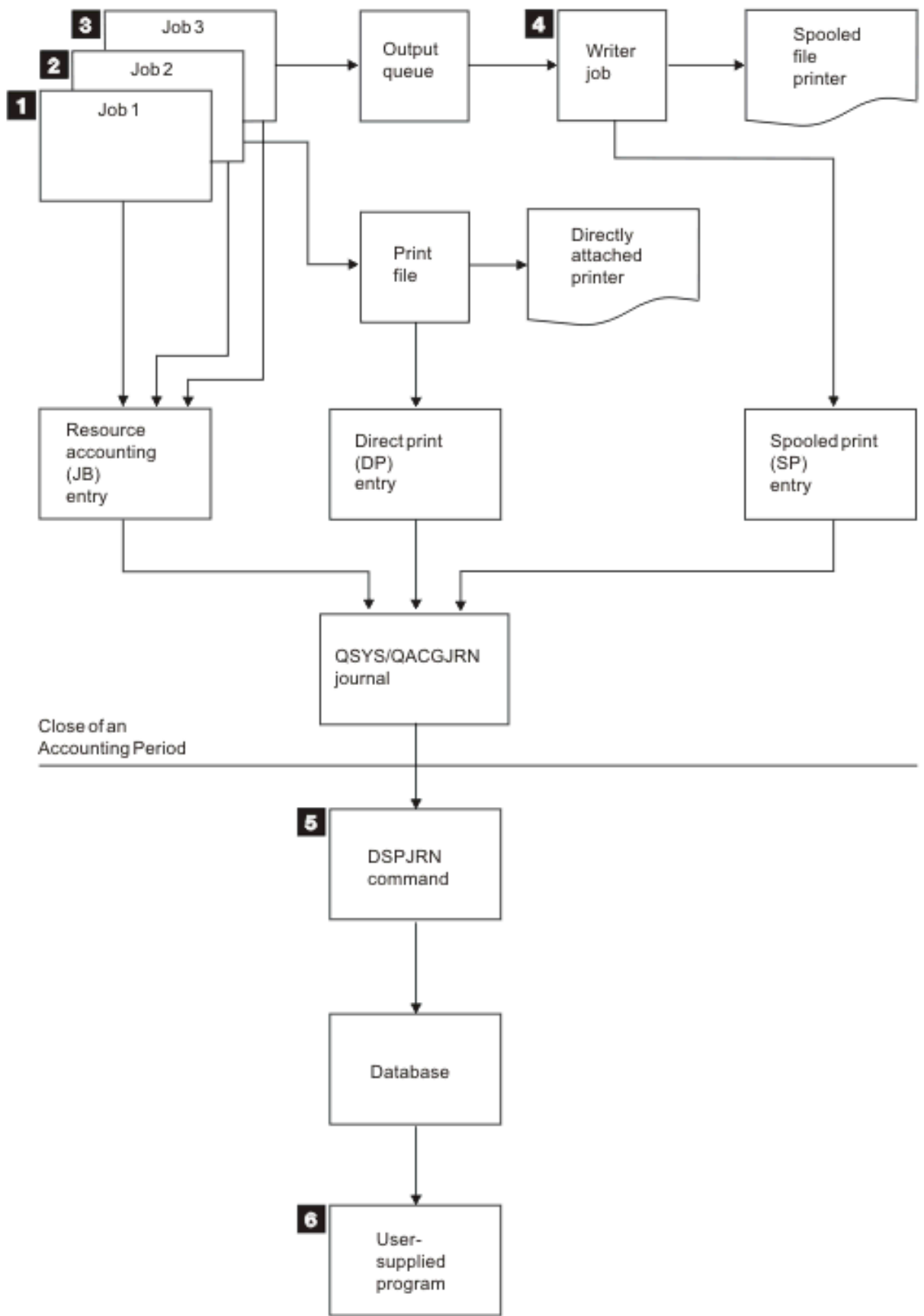


Figure 4. Job Accounting Overview

1. When Job1 is completed, the system summarizes the resources used and writes the JB journal entry to the QACGJRN journal. If the accounting code was changed during the job, a JB journal entry will be written for each time the accounting code was changed and at the end of the job. Job1 does not make any printer output, and no job log is made. Therefore, no direct print (DP) or spooled print (SP) journal entries are made for Job1.
2. Job2 is printing a file directly to a printer. When the file is completed a DP journal entry is written that summarizes the printed data. When Job2 is completed, the system summarizes the resources used and writes the JB journal entry. Job2 does not make any spooled printer output and no job log is made. Therefore, no SP journal entry is made for Job2.
3. Job3 is printing to a file that is spooled. The SP journal entry is not written unless a print writer prints the file. When Job3 is completed, the system summarizes the resources used and writes the JB journal entry. If a job log is made at the completion of the job, it is considered a normal spooled file and an SP journal entry is created if the file is printed.
4. A print writer is started and it prints the files made by one or more jobs. When the writer finishes a file, it makes an SP journal entry. The SP journal entry is not made if the file is canceled before printing starts.
5. At the close of an accounting period, the Display Journal (DSPJRN) command can be used to write the accumulated journal entries into the database file.
6. User-written programs or the query utility can be used to analyze the accounting data. Reports such as resources used will compile data by a specific accounting code, user, or job type.

### ***Job Accounting operating characteristics***

Your system attempts to allocate main storage as efficiently as possible. A job might not use the same amount of resources each time it is run.

For example, if there are several active jobs on your system, a job spends more time reestablishing the resources needed for running than if a dedicated system environment is used. The system uses the job and run priorities assigned to different jobs to assist in managing main storage. Therefore, high priority jobs can use less system resource than low priority jobs.

Because of these system operating characteristics, you might want to apply your own interpretation or algorithm to the job accounting data collected. If you are charging for the use of your system you might want to charge more for high priority jobs, work done during peak system time, or the use of critical resources.

### ***Accounting Journal Processing***

The accounting journal QSYS/QACGJRN is processed as any other journal. Files can also be recorded in this journal although for simplicity it is recommended that you keep it solely for accounting information.

You can use the Send Journal Entry (SNDJRNE) command to send other entries to this journal. While there are additional operational considerations involved in using several journals, there are advantages to *NOT* allowing any file entries in the QACGJRN journal. It is typically easier to control the QACGJRN journal separately so that all job accounting entries for a particular accounting period are in a minimal number of journal receivers and that a new journal receiver is started at the beginning of an accounting period. System entries also appear in the journal QACGJRN. These are the entries with a journal code of J, which relate to IPL and general operations performed on journal receivers (for example, a save of the receiver).

### ***Job accounting entries***

Job accounting entries are placed in the journal receiver starting with the next job that enters the system after the Change System Value (CHGSYSVAL) command takes effect. The accounting level of a job is determined when it enters the system. If the Journal accounting information (QACGLVL) system value is changed after the job is started, it has no effect on the type of accounting being performed for that job. The direct print (DP) and spooled print (SP) entries occur if the job that created the file is operating under accounting and the system value is set for \*PRINT. If spooled files are printed after the accounting level has been set to \*PRINT or if the job that created the file was started before the accounting level was changed, no journaling is done for those spooled files.

## When to use job accounting

These methods help you determine whether you should use job accounting and when to use job accounting.

### Additional information supplied by job accounting

Job accounting has all the information supplied by CPF1164 plus:

- Accounting code
- Number of print files, lines, and pages created by programs
- Number of database read, write, and update operations
- Number of communications read and write operations
- Actual lines and pages printed
- Time the job was active and suspended
- Actual number of bytes of control information and print data sent to the printer

### The job accounting function is more effective for gathering job accounting statistics if:

- The resource information regarding database, printer, and communications use is important.
- Accounting codes are assigned to users or jobs.
- The information for printed output is important.
- Job accounting must be done on an accounting segment basis in a job rather than on a complete job basis.
- The active and suspended time information is needed.

**Note:** Some statistics recorded in the CPF1164 message and the JB journal entries will not match exactly. This is due mainly to two factors: (1) CPF1164 statistics are recorded slightly before the JB journal statistics and (2) each time an accounting code is changed, rounding occurs for some fields, while rounding occurs only once for CPF1164 messages.

## Security and job accounting

Only the security officer (or a program adopting his authority) or a user with \*ALLOBJ and \*SECADM authority can change the Journal accounting information (QACGLVL) system value.

The change takes effect when a new job enters the system. This restriction ensures that if job accounting is in effect and the security officer performs a system IPL, an accounting entry is written for the security officer's job.

### Authority to assign job accounting codes

You can assign job accounting codes only if you have the authority to use the Create User Profile (CRTUSRPRF), Change User Profile (CHGUSRPRF) or Change Accounting Code (CHGACGCDE) command. This restricts the use of accounting codes and provides a basis for validity checking any changes.

Only a user with the \*SECADM special authority is allowed to use the CRTUSRPRF and CHGUSRPRF commands. However, the security officer can delegate this authority by creating a CL program, which allows another user to adopt the security officer's profile and change the ACGCDE parameter in the user profile. The individual could then have authority to one or more CL programs.

The ACGCDE parameter also exists in job description objects, but you must have the authority to use the CHGACGCDE command to enter a value other than the default of \*USRPRF. CHGACGCDE is shipped with PUBLIC authority of \*USE.

## Authority to CHGACGCDE Command

If you allow a user to use the Change Accounting Code (CHGACGCDE) command, that user can:

- Create or change the ACGCDE parameter in job descriptions. (Authority to create or change job descriptions is also required.)
- Change the accounting code in his current job.
- Change the accounting code of a job other than his own if he also has the \*JOBCTL special authority.

You can provide additional security by using the CHGACGCDE command in a CL program, which adopts the program owner's authority. This allows the user who is running an external function to perform a security-sensitive function without having direct authorization to the CHGACGCDE command.

The accounting journal and its receivers are treated as any other journal objects from a security viewpoint. You must decide what authorization should exist for the accounting journal and journal receiver.

### Related tasks

[Controlling the assignment of accounting codes](#)

An important aspect of any data processing application is ensuring that the correct control fields are specified. For job accounting codes, this can require a complex validity-checking function that not only checks for the existence of authentic codes, but also checks which users are allowed to use specific codes.

## About the accounting code

The initial accounting code (up to 15 characters in length) for a job is determined by the value of the ACGCDE (accounting code) parameter in the job description and user profile for the job.

When a job is started, a job description is assigned to the job. The job description object contains a value for the ACGCDE parameter. If the default of \*USRPRF is used, the accounting code in the job's user profile is used.

**Note:** When a job is started using the Submit Job (SBMJOB) command, its accounting code is the same as that of the submitter's job.

You can change the accounting code after the job has entered the system by using the Change Accounting Code (CHGACGCDE) command.

The CRTUSRPRF and CHGUSRPRF commands support the ACGCDE parameter. The default is \*BLANK. If all work for a particular user is to be recorded under one accounting code, only user profiles need to be changed. You can change the accounting codes for specific job descriptions by specifying the accounting code for the ACGCDE parameter on the CRTJOB and CHGJOB commands. The CHGACGCDE command also allows different accounting codes in a single job.

The Retrieve Job Attributes (RTVJOB) command and the API's that retrieve job attributes allow you to access the current accounting code in a CL program.

### Related tasks

[Setting up job accounting](#)

To set up job accounting, use the character based interface.

[Controlling the assignment of accounting codes](#)

An important aspect of any data processing application is ensuring that the correct control fields are specified. For job accounting codes, this can require a complex validity-checking function that not only

checks for the existence of authentic codes, but also checks which users are allowed to use specific codes.

## Resource accounting

Job resource accounting data is summarized in the job (JB) journal entry at the completion of a job. In addition, the system creates a JB journal entry summarizing the resources used each time a Change Accounting Code (CHGACGCDE) command occurs. The JB journal entry includes:

- Fully qualified job name
- Accounting code for the accounting segment just ended
- Processing unit time
- Number of routing steps
- Date and time the job entered the system
- Date and time the job started
- Total transaction time (includes service time, ineligible time, and active time)
- Number of transactions for all interactive jobs
- Auxiliary I/O operations
- Job type
- Job completion code
- Number of printer lines, pages, and files created if spooled or printed directly
- Number of database file reads, writes, updates, and deletes
- Number of ICF file read and write operations

**Note:** Some of the job accounting information can also be accessed using the CPF1124 and CPF1164 messages located in the QHST log.

## Resource accounting data

When analyzing the journal entries, it is important to understand how and when journal entries are written. A JB journal entry is written to the job accounting journal for a job any time the job accounting code is changed and when the job ends. Therefore, one job may have multiple journal entries.

Each resource accounting journal entry contains information about the resources used while the previous accounting code was in effect. Consider the following example:

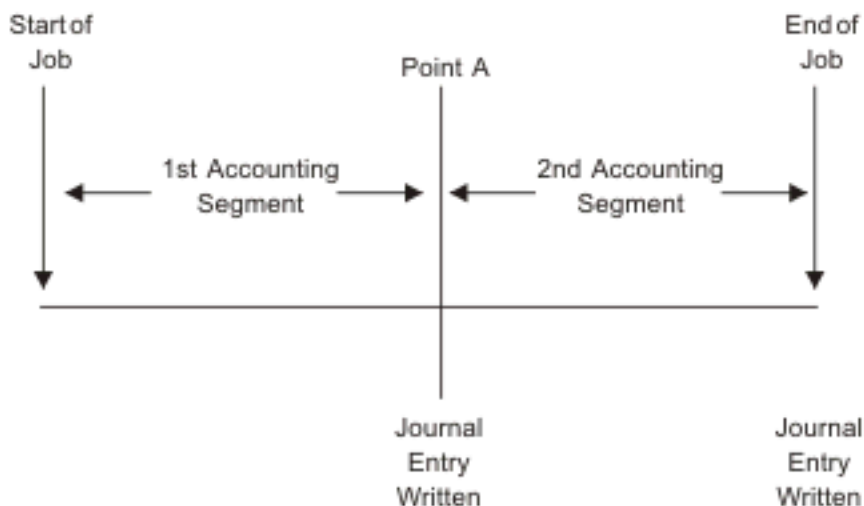


Figure 5. Resource accounting data example



At point A, the CHGACGCDE command was issued. The accounting code is changed and the JB journal entry is sent to the journal. The JB journal entry contains data for the first accounting segment. When the job ends, a second JB entry is made for the job containing data for the second accounting segment.

If the job accounting code was not changed during the existence of the job, the single JB entry summarizes the total resources used by the job. If the job accounting code was changed during the existence of the job, then you must add up the fields in the multiple JB entries in order to determine the total resources used by the job. The creation of a job log does not count toward the processing unit use for a job or its printed output in the JB accounting entries. However, if you are using print file accounting, the job log printed is included in the printer file journal entries.

## Prestart communications jobs and job accounting

If your system uses job accounting, the prestart job program should run the Change Prestart Job (CHGPJ) command with the program start request value for the accounting code parameter (CHGPJ ACGCDE(\*PGMSTRRQS)) immediately after the program start request attaches to the prestart job.

This action changes the accounting code to the value specified in the user profile associated with the program start request. Immediately before the program finishes handling the program start request, the program should run the Change Prestart Job command (CHGPJ) with the Prestart Job Entry value for the accounting code parameter (CHGPJ ACGCDE(\*PJE)). This changes the accounting code back to the value specified in the job description of the prestart job entry.

## Prestart jobs for batch applications

Prestart jobs and server jobs that use prestart jobs are usually configured to start with a generic user profile such as QUSER, and then they wait for a request to be handled. When a prestart job is given a request to handle, the job swaps user profiles using the Set Profile Handle (QWTSETP) API to that of the requester, services the request, and then swaps back to the initial user profile. If the prestart job is configured to be reused (MAXUSE parameter on the Add Prestart Job Entry (ADDPJE) or the Change Prestart Job Entry (CHGPJE) command is greater than 1) the job will wait for another request and repeat the above scenario. In this case, a single prestart job could potentially service many different users. If you want to be able to charge each of these users for their resources used, the accounting code needs to be updated before and after each service request. System-defined server jobs already do this for you.

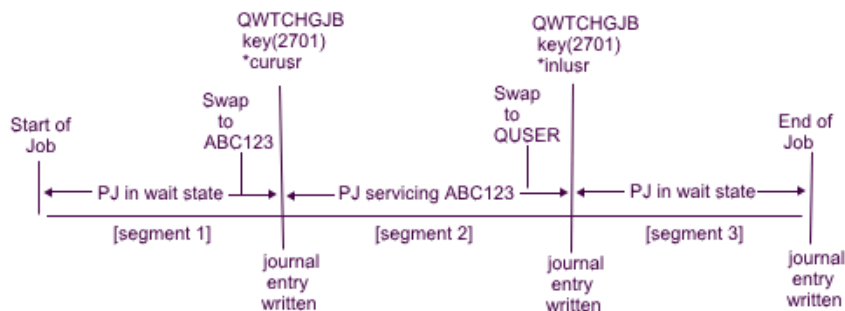


Figure 3. Prestart job with 3 accounting segments

The following is what the three journal entries, in the above picture would look like if a SQL or query was used to format:

Table 4. Prestart job with three accounting segments

Journal Entry #	Job Name	Job User	Job Number	User Profile	Accounting Code	CPU	Transactions
1	QSVREX1	QUSER	123456	ABC123	QUSER	50	1
2	QSVREX1	QUSER	123456	QUSER	ABC123	3729	120
3	QSVREX1	QUSER	123456	QUSER	QUSER	73	2

The resources used, such as CPU and transactions, can be charged back to the accounting code, but not necessarily to the user listed under the User Profile field (JAUSPF). The user profile is the current user at the time the journal entry is written, but it is not necessarily the user profile that was active during the entire accounting segment. In this example, the user profile has been swapped once in each of the first two segments. Since the journal entry is written after the swap, the current user profile logged in the entry is not the user who used the resources during the prior accounting segment.

Likewise, the Job User cannot reliably be used to charge for resources used, because this is the user that the job started with, and as part of the qualified job name, it does not change, even when servicing a different user. The accounting code is the only reliable field that can be used for charging resource usage. The accounting code differs from the other user fields because the accounting code is saved with the job until it is changed. At the time of the change, the job's current accounting code is written to the journal entry first, and then the new accounting code is stored in the job.

### Related concepts

#### Prestart jobs

A prestart job is a batch job that starts running before a work request is received. The prestart jobs are started before any other types of jobs in a subsystem. Prestart jobs are different from other jobs because they use prestart job entries (part of the subsystem description) to determine which program, class, and storage pool to use when they are started.

#### Managing prestart jobs

You can use prestart jobs to reduce the amount of time required to handle a program start request. These are the most common tasks associated with prestart jobs that you can perform.

### Related information

[Experience Report: Tuning prestart job entries](#)

[Experience Report: Job Accounting](#)

## System job processing for job accounting

System jobs that you control (for example, readers and writers) are assigned an accounting code of \*SYS. Other system jobs that you do not control (for example, QSYSARB, QCLUS, SCPF) do not receive a journal entry.

**Note:** You cannot use the Change Accounting Code (CHGACGCDE) command to change the accounting code of the subsystem monitor or a reader or writer. You can, however, change the accounting code of a reader or writer by changing the appropriate IBM-supplied job descriptions and user profiles and then starting them again.

## Batch processing and job accounting

Any batch job that is submitted using the Submit Job (SBMJOB) command automatically uses the same accounting code as the job that submitted the batch job. When the SBJOB command is used, the accounting codes cannot be overridden regardless of how the job description entry is coded.

If you want the batch job to operate under an accounting code other than that of the submitting job, a Change Accounting Code (CHGACGCDE) command should be issued either:

- Before and after the SBJOB command is issued
- Immediately by the batch job.

Batch jobs submitted using a reader or a Submit Database Job (SBMDBJOB) command use the accounting code specified in the job description for the batch job. If the job description specifies ACGCDE(\*USRPRF), the accounting code is taken from the user profile used for the job.

## Interactive processing and job accounting

If an interactive job has a fixed set of options for a user and each option has an assigned accounting code, it might be desirable to automatically assign a new code when the user requests to work on a new function.

A typical approach is for a menu option to request a new functional area. The Change Accounting Code (CHGACGCDE) command is then issued within a CL program and the job values used for the previous accounting code are summarized in the JB accounting journal entry.

If a user has several assignments for which only the user knows the accounting codes to be used you can:

- Give authority to the user to enter the CHGACGCDE command.
- Write a program to prompt the user for the accounting code.

**Note:** For source pass-through jobs, the job accounting information does not include the target pass-through job. For target pass-through jobs, the job accounting information does not include the associated communications batch job.

## Printer file accounting

There are two types of journal entries for printer file accounting; DP for nonspooled printer files and SP for spooled printer files. These two types of journal entries share a common journal entry format although some of the information is only available in the SP entry. The DP and SP journal entries include information such as:

- Fully qualified job name
- Accounting code
- Device file name and library
- Device name
- Device type and model
- Total number of pages and lines printed. If multiple copies occurred, this is the sum of all copies
- Spooled file name (only in the SP entry)
- Spooled file number (only in the SP entry)
- Output priority (only in the SP entry)
- From type (only in the SP entry)
- Form type (only in the SP entry)
- Total number of bytes of control information and print data sent to the printer device. If multiple copies occurred, this is the sum of all copies. (This only applies to the SP entry.)

The DP and SP journal entries occur when the file is printed. If a spooled file is never printed, no SP journal entry will appear.

## Journal entries for job accounting

The system provides different journal entries for the different types of data that can be gathered:

- Job resource accounting: The job (JB) journal entry contains data summarizing the resources used for a job or for different accounting codes used in a job.
- Printer file accounting:
  - Direct print (DP) journal entry: Contains data about printer files produced on print devices (nonspooled).
  - Spooled print (SP) journal entry: Contains data about printer files made by a print writer (spooled).

## Job accounting journal entry field information

These tables list the fields information that are in the job journal entry. Additional information about the various fields is found in the field reference files QSYS/QAJBACG and QSYS/QAJBACG4.

Field Name (Character 14)	Description	Field Attributes	Comments
JAJOB	Job name	Character (10)	
JAUSER	Job user	Character (10)	
JANBR	Job number	Zoned (6,0)	
JACDE	Accounting code	Character (15)	
JACPU	Processing unit time used (in milliseconds)	Packed decimal (11,0)	The processing unit time does not include processing unit use and printer statistics for the creation of job logs.
JARTGS	Number of routing steps	Packed decimal (5,0)	
JAEDTE	Job entered the system - Job entry date (mmddy format)	Character (6)	
JAETIM	Job entered the system - Job entry time (hhmmss format)	Character (6)	
JASDTE	Job start date and time - Job start date (mmddy format)	Character (6)	For job completion date and time from journal entries, use the JODATE and JOTIME fields that are part of the standard journal entry prefix information. (See the Backup and Recovery book for more information about these fields.) After an abnormal system ending, these fields contain the current date and time and not (as with CPF1164 messages) the actual time of the system ending.
JASTIM	Job start date and time - Job start time (hhmmss format)	Character (6)	For job completion date and time from journal entries, use the JODATE and JOTIME fields that are part of the standard journal entry prefix information. (See the Backup and Recovery book for more information about these fields.) After an abnormal system ending, these fields contain the current date and time and not (as with CPF1164 messages) the actual time of the system ending.
JATRNT	Total transaction time (in seconds)	Packed decimal (11,0)	The total transaction time is set to -1 when: <ul style="list-style-type: none"> <li>• Time is set backward.</li> <li>• An overflow occurred in a file on a computation.</li> <li>• The system went down while the job was active.</li> </ul>

Table 5. Job journal entry fields (continued)

Field Name (Character 14)	Description	Field Attributes	Comments
JATRNS	Number of transactions	Packed decimal (11,0)	The last transaction (SIGNOFF) is not counted.
JAAUX	Synchronous auxiliary I/O operations and database operations (including page faults for any reason)	Packed decimal (11,0)	
JATYPE	Job type	Character (1)	<p>The job types recorded are the following:</p> <p><b>A</b> Autostart job  <b>B</b> Batch job (includes communications and MRT)  <b>I</b> Interactive job  <b>M</b> Subsystem monitor  <b>R</b> Spooling reader  <b>W</b> Spooling writer</p> <p><b>Note:</b> These are the same as those used in message CPF1164, except that message CPF1164 includes some system job information not included in the journal entries.</p>
JACCDE	Completion code	Packed decimal (3,0)	<p>The completion codes, which are similar to those used for message CPF1164, are:</p> <p><b>000</b> Normal completion  <b>010</b> Normal completion during controlled end or controlled subsystem end  <b>020</b> Job exceeded end severity  <b>030</b> Job ended abnormally  <b>040</b> Job ended before becoming active  <b>050</b> Job ended while active  <b>060</b> Subsystem ended abnormally while job was active  <b>070</b> System ended abnormally while job was active  <b>080</b> Job completed in the time limit  <b>090</b> Job forced to complete after the time limit has ended  <b>099</b> Accounting entry caused by CHGACGCDE command</p>

Table 5. Job journal entry fields (continued)

Field Name (Character 14)	Description	Field Attributes	Comments
JALINE	Number of print lines	Packed decimal (11,0)	The number of print lines does not reflect what is actually printed. Spooled files can be canceled or printed with multiple copies. The information in the JB journal entry reflects only what was written by the program. This excludes any lines written for the job log. See the discussion on DP and SP printer file accounting data later in this chapter.
JAPAGE	Number of printed pages	Packed decimal (11,0)	
JAPRTF	Number of print files	Packed decimal (11,0)	
JADBPT	Number of database write operations	Packed decimal (11,0)	The numbers recorded for database I/O operations do not include I/O operations to readers and writers, or I/O operations caused by the CL commands CPYSPLF, DSPSPLF, or WRKSPLF. If SEQONLY(*YES) is in effect, these numbers show each block of records read, not the number of individual records read.
JADBGT	Number of database read operations	Packed decimal (11,0)	The numbers recorded for database I/O operations do not include I/O operations to readers and writers, or I/O operations caused by the CL commands CPYSPLF, DSPSPLF, or WRKSPLF. If SEQONLY(*YES) is in effect, these numbers show each block of records read, not the number of individual records read.
JADBUP	Number of database update, delete FEOD, release, commit, and rollback operations	Packed decimal (11,0)	The numbers recorded for database I/O operations do not include I/O operations to readers and writers, or I/O operations caused by the CL commands CPYSPLF, DSPSPLF, or WRKSPLF. If SEQONLY(*YES) is in effect, these numbers show each block of records read, not the number of individual records read.
JACMPT	Number of communications write operations	Packed decimal (11,0)	The numbers recorded for communications I/O operations do not include remote workstation activity. When the I/O is for a communications device, the numbers include only activity related to ICF files.
JACMGT	Number of communications read operations	Packed decimal (11,0)	The numbers recorded for communications I/O operations do not include remote workstation activity. When the I/O is for a communications device, the numbers include only activity related to ICF files.
JAACT	Time job was active (in milliseconds)	Packed decimal (11,0)	

Table 5. Job journal entry fields (continued)

<b>Field Name (Character 14)</b>	<b>Description</b>	<b>Field Attributes</b>	<b>Comments</b>
JASPN	Time job was suspended (in milliseconds)	Packed decimal (11,0)	
JAEDTL	Timestamp job entered system (mmddyyyyhhmmss)	Character (14)	
JAESTL	Timestamp job started (mmddyyyyhhmmss)	Character (14)	
JAAIO	Asynchronous I/O for database and non-database operations	Packed decimal (11,0)	
JAXCPU	Expanded CPU time used	Packed decimal (29,0)	
JAXSIO	Expanded synchronous auxiliary I/O operations	Packed decimal (29,0)	
JAXAIO	Expanded asynchronous auxiliary I/O operations	Packed decimal (29,0)	
JAXDBP	Expanded number of database puts	Packed decimal (29,0)	
JAXDBG	Expanded number of database gets	Packed decimal (29,0)	
JAXDBU	Expanded number of database updates and deletes	Packed decimal (29,0)	
JAXLIN	Expanded number of lines printed	Packed decimal (29,0)	
JAXPAG	Expanded number of pages printed	Packed decimal (29,0)	
JAXPRT	Number of print files	Packed decimal (29,0)	

**Printer file accounting data for direct print and spooled print**

The accounting code used for the direct print (DP) or spooled print (SP) journal entries is the accounting code of the job at the time the file is closed. Sometimes a DP or SP entry is created before the file is closed (such as when a writer which is creating a SCHEDULE(\*IMMED) file is ended). When this happens the current accounting code of the job is used.

A DP or an SP journal entry is created for each file printed. If the job log is spooled and then printed, an SP entry is created for it. Also, an SP entry is written for diskette spooled files redirected to a printer by the print writer.

*DP accounting journal information*

The file QSYS/QAPTACG5 contains fields that are used in the direct print (DP) journal entry. This table lists these fields and their attributes.

*Table 6. Direct print journal entry fields*

Field Name	Description	Field Attributes
JAJOB	Job name	Character (10)
JAUSER	Job user	Character (10)
JANBR	Job number	Zoned (6,0)
JACDE	Accounting code	Character (15)
JADFN	Device file name	Character (10)
JADFNL	Library in which device file is stored	Character (10)
JADEVN	Device name	Character (10)
JADEVT	Device type	Character (4)
JADEVM	Device model	Character (4)
JATPAG	Total number of print pages produced	Packed decimal (11,0)
JATLIN	Total number of print lines produced	Packed decimal (11,0)
JASPFN	Always blank	Character (10)
JASPNB	Always blank	Character (4)
JAOPTY	Always blank	Character (1)
JAFMTP	Always blank	Character (10)
JABYTE	Always zero	Packed decimal (15,0)
JAUSRD	User data	Character (10)
JALSPN	Always blank	Character (6)
JASPSY	Always blank	Character (8)
JASPDT	Always blank	Character (7)
JASPTM	Always blank	Character (6)
JADFASP	Always blank	Character (10)

*SP accounting journal information*

This table lists the fields (found in file QSYS/QAPTACG5) that are used in the spooled print (SP) journal entry.

**Note:** SP accounting journal information is similar to that provided in the direct print (DP) accounting journal data except that the spooled file name, spooled file number, output priority, form type, and total number of bytes of control information and print data sent to the printer are included. An SP journal entry is not written if a spooled file is deleted before a writer starts writing the file to the device.



Table 7. Spooled print journal entry fields

Field name	Description	Field attributes
JAJOB	Job name	Character (10)
JAUSER	Job user	Character (10)
JANBR	Job number	Zoned (6,0)
JACDE	Accounting code	Character (15)
JADFN	Device file name	Character (10)
JADFNL	Library in which device file is stored	Character (10)
JADVN	Device name	Character (10)
JADVT	Device type	Character (4)
JADVM	Device model	Character (4)
JATPAG	Total number of print pages produced	Packed decimal (11,0)
JATLIN	Total number of print lines produced	Packed decimal (11,0)
JASPFN	Spooled file name	Character (10)
JASPNB	Spooled file number	Character (4)
JAOPY	Output Priority	Character (1)
JAFMTP	Form type	Character (10)
JABYTE	Total number of bytes sent to the printer	Packed decimal (15,0)
JAUSRD	User Data	Character (10)
JALSPN	Spooled file number	Character (6)
JASPSY	Spooled file job system name	Character (8)
JASPDT	Spooled file create date (cyymmdd format)	Character (7)
JASPTM	Spooled file create time (hhmmss format)	Character (6)
JADFASP	ASP name for device file library	Character (10)

**Note:**

- The system attempts to record the actual number of pages, lines, and bytes printed, but when a writer is canceled \*IMMED or recovers from a device error (such as end of forms), it is not possible to determine the exact number of pages, lines, and bytes printed.
- Extra pages and lines produced with the alignment line are not included in the page, line, and byte counts.
- If a spooled file goes into WTR status (but is set to MSGW) or if the file is deleted while in MSGW status, an SP journal entry will appear in the DP accounting journal indicating that there are 0 pages and 0 lines printed.

- While using a printer configured AFP(\*YES), if you delete or hold a file immediately after it has printed pages, the SP entry for that file may indicate 0 pages and 0 lines printed although some pages were printed.
- The page, line, and byte counts for the job and file separators are included with the counts for the file they are associated with.
- When an IPDS file contains graphics or bar codes and is sent to an IPDS printer that does not support graphics or bar codes, the page, line, and byte counts include the graphics and bar codes that are not printed.
- If printer configuration is AFP(\*YES), the field for total number of print lines produced is zero. The total number of pages produced field is correct.

## Managing 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 about the different types of daily work management tasks as well as other tasks you might need to perform on your system. Each subtopic explains why it is important to do these tasks, as well as how to complete them.

### Calling a special IPL recovery program

To call a special recovery program for situations when the IPL senses that the previous system ending was abnormal, you can add an autostart job entry to the subsystem description for the controlling subsystem.

This program checks the Previous system ending status (QABNORMSW) system value. For a normal system ending, the value of QABNORMSW is '0', and for an abnormal system ending the value of QABNORMSW is '1'. An alternative is to drop the messages and start up other subsystems when your recovery function is complete.

```

1.00 /* SPCRECOV - Autostart program to call special recovery program */
2.00      PGM
3.00      DCL          &QABNORMSW *CHAR LEN(1)
4.00      RTVSYSVAL  SYSVAL(QABNORMSW) RTNVAR(&QABNORMSW)
5.00      IF          (&QABNORMSW *EQ '1') DO /* Recover */
6.00      SNDPGMMSG  MSG('Recovery program in operation-do not +
7.00                  start subsystems until notified') +
8.00                  TOMSGQ(QSYSOPR)
9.00      CALL        RECOVERY
10.00     SNDPGMMSG  MSG('Recovery complete-jobs may be started') +
11.00                  TOMSGQ(QSYSOPR)
12.00     ENDDO /* Recover */
13.00     ENDPGM

```

#### Related information

[Changing the IPL start-up program](#)

### Monitoring 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. You can accomplish this in a variety of ways, such as using IBM Navigator for i.

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

You can access the System Status window from the **System** folder.

To get to System Status from the **System** folder:

1. Expand **System**.

## 2. Click **System Status**.

For more information about the different tasks that you can complete using system status, see the IBM Navigator for i help.

### **Checking memory pool usage**

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 IBM Navigator for i, you can easily monitor the amount of memory your pools are using.

To check the memory use, follow the following steps:

1. From IBM Navigator for i, expand > **Work Management** > **Memory Pools** > **Active Memory Pools** or **Shared Memory Pools**.
2. Right-click the memory pool you want to work with (for example, Interactive) and click **Properties**.
3. Click the Configuration tab. The **Current** field, that is located within the Size group, shows the amount of memory that 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 that you see when a list of memory pools displays in the right pane.

## **Controlling levels of system activity**

You can control how much activity is on the system by controlling how many jobs can be active at the same time in a subsystem or by controlling the use of the processing unit by jobs that have already been started.

Table 8. Ways to control system activity levels

What can I control?	What can I use to control?	Character-based interface method	IBM Navigator for i interface method
Number of active jobs	Subsystem Description	<p>Command: CHGSBSD MAXJOBS</p> <p>Use this parameter to specify how many jobs can be active at the same time in a subsystem.</p> <p>For an active subsystem, the sum of all of the jobs that are active at the same time that are started through work entries in the subsystem cannot exceed the MAXJOBS parameter value.</p> <p>This excludes autostart jobs, which might temporarily cause the limit to be exceeded when the subsystem is started.</p>	<p>Use the Run Command window.</p> <p><b>Expand system &gt; Run Command</b></p> <p>Type the command CHGSBSD and then click <b>Prompt</b>.</p>
	Job Queue Entry	<p>Command: CHGJOBQE MAXACT</p> <p>Use this parameter to specify how many batch jobs from a job queue can be active at the same time in the subsystem.</p> <p>A MAXACT of 1 for a job queue forces jobs to be selected serially by job priority from a job queue.</p> <p>The MAXPTYn parameter is used to specify how many jobs can be active for a specified job priority.</p>	<p>Use the Run Command window.</p> <p><b>Expand system &gt; Run Command</b></p> <p>Type the command CHGJOBQE and then click <b>Prompt</b>.</p>
	Workstation Entry	<p>Command: CHGWSE MAXACT</p> <p>Use this parameter if the WRKSTNTYPE parameter is specified. This parameter specifies how many interactive jobs can be active at the same time in the subsystem for that entry.</p>	<p>Use the Run Command window.</p> <p><b>Expand system &gt; Run Command</b></p> <p>Type the command CHGWSE and then click <b>Prompt</b>.</p>
	Communications Entry	<p>Command: CHGCMNE MAXACT</p> <p>Use this parameter to specify how many communications batch jobs can be active at the same time for that entry.</p>	<p>Use the Run Command window.</p> <p><b>Expand system &gt; Run Command</b></p> <p>Type the command CHGCMNE and then click <b>Prompt</b>.</p>
	Routing Entry	<p>Command: CHGRTGE MAXACT</p> <p>Use this command to specify how many jobs can be active at the same time using a given routing entry.</p>	<p>Use the Run Command window.</p> <p><b>Expand system &gt; Run Command</b></p> <p>Type the command CHGRTGE and then click <b>Prompt</b>.</p>
	Prestart job entry	<p>Command: CHGPJE MAXJOBS</p> <p>Use this command to specify how many prestart jobs can be active at the same time for that entry.</p>	<p>Use the Run Command window.</p> <p><b>Expand system &gt; Run Command</b></p> <p>Type the command CHGPJE and then click <b>Prompt</b>.</p>

Table 8. Ways to control system activity levels (continued)

What can I control?	What can I use to control?	Character-based interface method	IBM Navigator for i interface method
Number of active jobs (continued)	System	The Maximum eligible threads (QMAXACTLVL) system value is used to specify how many threads can share main storage and processor resources at the same time. All active jobs (including system jobs) in all storage pools are controlled by QMAXACTLVL.	<b>Configuration and Service &gt; System Values &gt; Performance category &gt; Memory Pools tab &gt; Maximum eligible threads</b>
Use of processing unit and main storage	Base storage pools	The Base memory pool maximum eligible threads (QBASACTLVL) system value is used to specify how many threads can share the Base storage pool at the same time and to limit main storage contention.	<b>Configuration and Service &gt; System Values &gt; Performance category &gt; Memory Pools tab &gt; Base Memory pool: Maximum eligible threads</b>
	Shared pools	Command: WRKSHRPOOL Use this command to specify the activity level for shared pools	<b>Work Management &gt; All Tasks &gt; Memory Pools &gt; Shared Memory Pools &gt; right-click a shared pool &gt; Properties &gt; Configuration tab</b> and change the <b>Maximum eligible threads</b> field
	Private storage pools	Command: CHGSBSD POOLS Use this command to specify the activity level for user-defined main storage pools.	Use the Run Command window. <b>Expand system &gt; Run Command</b> Type the command CHGSBSD and then click <b>Prompt</b> .

### Examples: activity control relationships

These examples show the relationship of some of the activity controls. Assume the system activity level is 100 and the jobs are single-threaded.

#### Base memory pool example

Two subsystems, SBSA and SBSB, use the Base memory pool to run jobs. SBSA currently has two jobs running in this memory pool and SBSB has one. A job queue entry in the subsystem description for SBSB specifies that any number of jobs can be started. The activity level of the Base memory pool is 3. Therefore, only three jobs in the Base memory pool can compete for the processing unit at a time. However, all of the jobs are started.

#### Four jobs in a subsystem example

One autostart job, two workstation jobs, and one batch job (four jobs in all) are in subsystem SBSC. The MAXACT for SBSC is specified as 4. No matter what is specified for the MAXACT of the work entries, no other jobs can be started until at least one job completes running.

#### Batch subsystem MAXACT(1) example

Subsystem SBSE is a batch subsystem for which 1 is specified for MAXACT. Although the job queue entry does not specify MAXACT, the limit is one job because 1 is specified for MAXACT for the subsystem. Therefore, jobs are processed in job priority one at a time off the job queue.

## Determining the status of a job

Monitoring your jobs can 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.

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

1. From IBM Navigator for i, expand **Work Management** > **Active Jobs** or **Server Jobs**.

**Note:** You can see a job status from anyplace within the Work Management folder that you access jobs.

2. 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).

**Tip:** If you do not see the Detailed Status column, you can add it to the display by opening **Active Jobs** (or **Server Jobs**) and selecting **Actions** > **Columns**.

## Monitoring a subsystem

Because subsystems are important to the daily activity done on your system, it is important that you monitor the activity in your subsystems.

Within a subsystem description you can specify the number of jobs that can run at one time in the subsystem by setting the maximum active jobs value. As the amount of work on your system increases you might want to change the maximum active jobs value in your subsystem. The number that you supply here should be set so that the available resources are properly utilized. Increasing the number of active jobs without increasing the resources available can hurt performance on your system.

To check the maximum active jobs value of your subsystem, you can use either IBM Navigator for i or the character-based interface.

### *IBM Navigator for i*

1. Expand **Work Management** > **Active Subsystems**.
2. Right-click the subsystem which you want to monitor.
3. Select **Properties**.

**Note:** Make sure you set this option very carefully. If you set maximum active jobs value too high, you might make your system perform slowly. However, if you set your maximum active jobs too low, your work might start to bottleneck and slow performance.

### *Character-based interface*

**Command:** Display Subsystem Description (DSPSBSD)

Select Option 1: Operational attributes, to see the value for maximum jobs in the subsystem.

## Determining the number of subsystems using a memory pool

Subsystems are allocated a certain percentage of memory to run jobs. It is important to know how many different subsystems are pulling from the same memory pool. After you know how many subsystems are submitting jobs to a pool and how many jobs are running in a pool, you might want to reduce resource contention by adjusting the size and activity level of the pool.

### *IBM Navigator for i*

To monitor the number of subsystems that are using a memory pool, follow the following instructions:

1. Expand **Work Management** > **Memory Pools** > **Active Memory Pools** or **Shared Memory Pools**.
2. Right-click the memory pool that you want to work with and click **Subsystems**.

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

## Character-based interface

**Command:** Work with Subsystems (WRKSBS)

This command displays a list of all of the subsystems and their corresponding pools.

## Viewing job performance statistics

A job's performance is important to anyone that uses a IBM Navigator for i product 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 disk 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, use the following instructions:

1. From IBM Navigator for i, expand **Work Management > 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 window can be displayed from the Performance tab of a Job property window.

2. Right-click the job for which you want to display the performance statistics, and click **Details > Elapsed Performance Statistics**.

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.

## Viewing overall system status

IBM Navigator for i puts all information relating to system status in one place. This makes it easier for you to monitor how your system is performing, identify potential trouble areas, and quickly determine what action you need to take to improve performance.

The System Status window divides the overall system status into six specific areas:

### General

This is the CPU elapsed usage percentage, number of active jobs, address used percentage, system disk pool usage percentage, total jobs on system, percentages of permanent and temporary addresses used, total disk space, and system disk pool capacity.

### Jobs

This is the total number of jobs, number of active jobs, maximum number of jobs, and the number of active threads.

### Processors

This is the CPU elapsed usage percentage. (Depending upon your hardware configuration, you may also see additional information regarding the type of processor(s), number of processors, processing power, virtual processors, interactive performance, elapsed shared processor pool usage, and elapsed uncapped CPU capacity usage.)

### Memory

This is the total memory (main storage) on your system and a button that gives you access to the list of active memory pools on the system.



## **Disk Space**

This is the total disk space, the system disk pool capacity and usage, information about the temporary storage used, and buttons that give you access to more disk status, the list of disk pools on the system, and storage system values information.

## **Addresses**

This is the information about permanent and temporary addresses used, large (256 MB) permanent and temporary addresses used, and very large (4 GB) permanent and temporary addresses used.

To view general system status, use the following instructions:

From IBM Navigator for i, expand **System** and click **System Status**.

The System Status window appears. For more information about this window, see the IBM Navigator for i online help.

## **Checking disk status**

At times you may want to check on the performance of the disk units on your system, or view status information regarding them.

To view the Disk Status window, follow the following steps:

From IBM Navigator for i, expand **System** and click **Disk Status**.

The Disk Status window is displayed.

You can use the **Customize this View** > **Columns** option of the Disk Status window to view the following information:

- Amount Read (KB)
- Amount Written (KB)
- Percentage Busy
- Compression
- Disk Pool
- I/O Requests
- Percentage Used
- Protection Status
- Protection Type
- Read Requests
- Request Size (KB)
- Size (MB)
- Type
- Write Requests

## **Managing jobs**

As any work management administrator knows, managing jobs is more than placing jobs on hold and moving jobs from job queue to job queue. This topic discusses the most common job management tasks as well as some of the more involved tasks that can help improve your system's performance.

### **Common job tasks**

These are the most common tasks that you can perform with jobs. The instructions apply to both IBM Navigator for i (where available) and the character-based interface.

#### **Starting a job**

Interactive jobs are started when the user signs on to a workstation. You start prestart jobs and batch jobs by using IBM Navigator for i or the character based interface, depending upon the circumstances.

### *Starting a batch job that is waiting in the job queue*

Occasionally you might need to force a job to start immediately. While moving the job to a job queue that is not busy is the most efficient method to accomplish this, there are some other methods that you can use.

To start a batch job, first check the status of the job queue in which the job resides and determine if moving the job to another queue makes the most sense for your situation. (**Work Management > All Tasks > Job Queues > Active Job Queues or All Job Queues**)

If moving the job to another queue isn't feasible, you can place the running jobs on hold and then move the job that you need to start up in priority. However, use caution when using this method because the held jobs are still included in the maximum active job count.

To change the priority of the job and indicate when it should run, use the following instructions:

1. Right-click the job and click **Properties**.
2. On the Job Properties window, click the **Job Queue** tab.
3. Change the **Priority on job queue** to a higher priority (0 is the highest).
4. Set the **When to make job available to run** to either Now or specify the date and time.
5. Click **OK**.

### *Starting a prestart job*

Prestart jobs typically start at the same time the subsystem is started. You manually start a prestart job when all prestart jobs have been ended by the system due to an error or were never started during subsystem start up due to STRJOBS (\*NO) on the prestart job entry. To start a prestart job, use the character-based interface.

**Command:** Start Prestart Jobs (STRPJ )

The STRPJ command should not be used until the startup of the related subsystem is complete. To make sure that the necessary prestart job successfully starts, code a delay loop with a retry if the STRPJ command fails.

The number of prestart jobs that can be active at the same time is limited by the MAXJOBS attribute on the prestart job entry and by the MAXJOBS attribute for the subsystem. The MAXACT attribute on the communications entry controls the number of program start requests that can be serviced through the communications entry at the same time.

**Note:** If you specified \*NO on the STRJOBS attribute, no prestart jobs start for the prestart job entry when the subsystem starts. Running the STRPJ command does not cause the value of the STRJOBS parameter to change.

**Example:** This example starts prestart jobs for prestart job entry PJPGM in subsystem SBS1. Subsystem SBS1 must be active when this command is issued. The number of jobs started is the number specified in the INLJOBS value of prestart job entry PJPGM. The subsystem starts program PJPGM in library PJLIB.

```
STRPJ  SBS(SBS1)  PGM(PJLIB/PJPGM)
```

### **Ending a job**

You can use IBM Navigator for i or the character-based interface to end a job. The job can be active or on a job queue. You can end a job immediately or by specifying a time interval so that end of job processing can occur.

#### *IBM Navigator for i*

1. Expand **Work Management > Active Jobs**.
2. Locate the job that you want to end.
3. Right-click the job and click **Delete/End**.
4. Complete the Confirm Delete/End window and click **Delete**.

## Character-based interface

### **Command:** End Job (ENDJOB)

If you do not know the name of the job that you want to end, you can use one of the following commands to find the job name:

- Work with Active Jobs (WRKACTJOB)
- Work with User Jobs (WRKUSRJOB)
- Work with Submitted Jobs (WRKSBMJOB)
- Work with Subsystem Jobs (WRKSBSJOB)
- End Subsystem (ENDSBS) This command ends all of the jobs in the subsystem.
- End System (ENDSYS) This command ends most activity on the system and leaves the system in a condition in which only the console is active in the controlling subsystem.
- Power Down System (PWRDWN SYS) This command prepares the system for ending and then starts the power-down sequence.

A job may be ended either immediately or in a controlled manner. It is strongly recommended that you always attempt to end a job in a controlled manner.

### *Ending a job: controlled*

Ending a job in a controlled manner allows programs that are running in the job to perform their end-of-job cleanup. A delay time can be specified to allow the job to end in a controlled manner. If the delay time ends before the job ends, the job is ended immediately.

Any application that needs to perform end-of-job cleanup should detect when the job is ending in a controlled manner. There are three ways an application can detect this:

### **Synchronously retrieve End Status**

At certain points, an application can synchronously check the End Status of the job in which it is running. You can retrieve the job's end status by issuing the Retrieve Job Attributes (RTVJOB) CL command. Additionally, you can use one of several APIs that retrieve the job's end status. You can find more information about these APIs in the experience report, *Work management job attributes*

### **Synchronously check major and minor return codes after an I/O operation**

For both display I/O and ICF communications I/O, a major return code of 02, or a major return code of 03 with a minor return code of 09 indicates the job is ending in a controlled manner.

### **Handle the asynchronous signal SIGTERM**

Some applications use a signal handling program to improve the cleanup of the application when the job is ended. The system generates the asynchronous signal SIGTERM for the job being ended, when the job is ending controlled and all of the following conditions are met:

- The job is enabled for signals
- The job is a signal handling program that is established for the SIGTERM signal
- The job is currently running in the problem phase

If any of the above conditions are not met, the SIGTERM signal is not generated for the job being ended.

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 controlled manner.

### **Related tasks**

[Stopping a subsystem](#)

You can use IBM Navigator for i or the character-based interface 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.

### **Related information**

[Jobs system values: Maximum time for immediate end](#)

#### *Ending a job: immediate*

When a job ends immediately, you can get undesirable results such as application data that has been partially updated. Use the immediate end option only if a controlled end has been unsuccessful.

Before ending the job, you should verify that no logical unit of work is in an in doubt state due to a two-phase commit operation that is in progress. If it is, then the value of the Action ifENDJOB commitment option can greatly impact the ENDJOB processing. This option is part of the Change Commitment Options (QTNCHGCO) API. For example, if the Action ifENDJOB commitment option is the default value of WAIT, this job is held up and does not complete its end of job processing until the commitment control operation is completed. This ensures database integrity on all related systems.

When you use the immediate end option, the system performs minimal end-of-job processing, which can include:

- Closing the database files
- Spooling the job log to an output queue
- Cleaning up internal objects in the operating system
- Showing the end-of-job display (for interactive jobs)
- Completing commitment control processing

### **Related information**

[Change Commitment Options \(QTNCHGCO\) API](#)

### ***Finding jobs***

It is important to understand how to find jobs on your system. Whatever the reason, at some point in time you might need certain information from a particular job.

In IBM Navigator for i, 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. 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 a large number 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 Filter and Include functions throughout work management wherever 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 open the list that you want to search before you can use these tools.

#### *IBM Navigator for i*

To find a job using the **Filter** option, use the following instructions:

1. Expand **Work Management > Active Jobs**.
2. In the **Filter** field, type the job ID you want to find (for example, Qqqtemp1). All the job columns are searched for your job.
3. Your job will be listed once it is found.

**Note:** Job names are not case sensitive.

#### *Limit information that is displayed*

To limit the information that is displayed, use the Include function.

1. Expand **Work Management** > **Active Jobs** or **Server Jobs**.
2. Click **Actions** > **Include**. The Include window is shown.
3. In the Include window, select the options with which you want to search for your job.
4. Click OK.

#### *Character-based interface*

To find a job on the system, use either the Work with Active Job (WRKACTJOB), Work with User Job (WRKUSRJOB), or Work with Submitted Job (WRKSBJOB) command.

### **Viewing 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.

#### *IBM Navigator for i*

To view jobs on the job queue, use the following instructions:

1. Expand **Work Management** > **All Tasks** > **Job Queues** > **Active Job Queues** or **All Job Queues**.
2. Click the job queue with which you want to display the jobs (for example, Jobqueue1). The jobs within the job queue appear.

#### *Character-based interface*

**Command:** Work with Job Queue (WRKJOBQ)

This command displays a list of all of the job queues available on the system. After you have located the job queue that contains your job you can select option **5=Work with** and display all of the jobs in the job queue.

You can also use the Work with Subsystems Job command to display a list job queues and their respective jobs.

**Command:** Work with Subsystem Job (WRKSBSJOB) SBS(\*JOBQ)

### **Viewing jobs in the subsystem**

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

#### *IBM Navigator for i*

To view jobs in the subsystem, follow these steps:

1. Expand **Work Management** > **Active Subsystems**.
2. Click the subsystem that has the jobs that you want to display.

#### *Character-based interface*

**Command:** Work with Active Jobs (WRKACTJOB SBS (subsystem name))

**Command:** Work with Subsystem Descriptions (WRKSBSD)

Use the Work with Subsystem Descriptions command to display a list of subsystems. After you find the subsystem that contains your job, use option **8=Work with subsystem jobs** to display the job information.

**Note:** The subsystem must be active to display the job information.

### **Viewing job attributes**

Job attributes contain information about how jobs are processed. They are originally specified when the job is created. Some of the attributes come from the job description. After the job is created, the job attributes can be viewed and managed through work management in IBM Navigator for i. The job properties pages make a system operator's job easier by providing efficient and easy-to-use functions for managing jobs.

## Related information

[Experience report: Work management job attributes](#)

*IBM Navigator for i*

To view job attributes, use the following instructions:

1. Expand **Work Management > Active Jobs** or **Server Jobs**, depending on the type of job that you want to work with.
2. Find the job whose properties you want to view or change.
3. Right-click the **Job Name** and click **Properties**.

Job attributes 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. Attributes for system jobs cannot be changed in IBM Navigator for i. However, the run priority of some system jobs can be changed in the character based interface using the Change System Job (CHGSYSJOB) command.

*Character-based interface*

**Command:** Work with Job (WRKJOB) When the job is active you can view the following information: job run attributes, call stack information, job lock information, library list information, job log information, open file information, file override information, commitment control status, communications status, activation group information, mutex information, and thread information

**Command:** Display Job (DSPJOB)

This command displays the following information about the job: job status attributes, job definition attributes, job run attributes, spooled file information, job log information, call stack information, job lock information, library list information, open file information, file override information, commitment control status, communications status, activation group information, mutex information, thread information, media library, and attribute information.

## Viewing call stacks

You can view information about a job or thread's call stack by using either IBM Navigator for i or the character-based interface.

### Related concepts

Call stacks

The *call stack* is the ordered list of all programs or procedures currently running for a job. The programs and procedures can be started explicitly with the CALL instruction, or implicitly from some other event.

*IBM Navigator for i*

1. Expand **Work Management > Active Jobs** or **Server Jobs**, depending on the type of job that you want to work with.
2. Right-click the job name and then click **Details > Call Stack**.

If you want to view a call stack for a thread, follow these steps:

1. Expand **Work Management > Active Jobs** or **Server Jobs**, depending on the type of job that you want to work with.
2. Right-click the job name, and then click **Details > Threads**.
3. From the list of threads, right-click a specific thread, and then click **Details > Call Stack**.

If you are running under a user profile with \*SERVICE special authority and want to see additional entries for LIC and IBM i PASE Kernel, from the Call Stack window, use the Include option from the Customize this view window. (**Include**)

*Character-based interface*

**Command:** Work with Jobs (WRKJOB) or Display Jobs (DSPJOB)

Select option 11: Display call stack, if active.

If you wish to see a call stack for a thread, after issuing the WRKJOB or the DSPJOB command, select option 20: Work with threads, if active. Then, select option 10: Display call stack option for the selected thread.

### ***Placing a job on the job queue***

Jobs are placed on the job queue by either moving an existing job from one queue to another, or by submitting a new job. Use IBM Navigator for i to move jobs between queues. Use the character-based interface to submit a new job.

#### *IBM Navigator for i*

To use the IBM Navigator for i interface, the job must already exist in another job queue. Then you can move the job from one queue to another queue. (To place a new job on a job queue, use the command line interface.)

1. Expand **Work Management > All Tasks > Job Queues > All Job Queues**.
2. Right-click the job queue that contains the job and select **Jobs**.
3. Right-click the job that you want to move.

The Move window opens where you can specify the destination queue.

#### *Character-based interface*

The following is a list of character-based interface methods for placing a new job on the new job queue.

- **Submit Job (SBMJOB)**: Allows a job that is running to submit another job to a job queue to be run later as a batch job. Only one element of request data can be placed on the new job's message queue. The request data can be a CL command if the routing entry used for the job specifies a CL command processing program (such as the IBM-supplied QCMD program).
- **Add Job Schedule Entry (ADDJOBSCDE)**: Automatically the system submits a job to the job queue at the time and date specified in the job schedule entry.
- **Submit Database Jobs (SBMDBJOB)**: Submits jobs to job queues so they can be run as batch jobs. The input stream is read either from a physical database file or from a logical database file that has a single-record format. This command allows you to specify the name of this database file and its member, the name of the job queue to be used, and to decide whether jobs being submitted can be displayed by the Work with Submitted Jobs (WRKSBMJOB) command.
- **Start Database Reader (STRDBRDR)**: Read a batch input stream from a database and place one or more jobs on job queues.
- **Transfer Job (TFRJOB)**: Move the current job to another job queue in an active subsystem.
- **Transfer Batch Job (TFRBCHJOB)**: Move the current job to another job queue.

### ***Moving a job to a different job queue***

There are many reasons why you might want to move a job to another queue. For example, sometimes jobs become backlogged in the queue because of a long running job. Perhaps the job's scheduled run time conflicts with a new job that has a higher priority. One way to manage this situation is to move the waiting jobs to another queue that is not as busy.

You can use either the IBM Navigator for i interface or the character-based interface to move a job from one queue to another.

#### *IBM Navigator for i*

1. Expand **Work Management > All Tasks > Job Queues > All Job Queues**.
2. Right-click the job queue that contains the job and select **Jobs**.
3. Right-click the job that you want to move.

The Move window opens where you can specify the target queue.

- Jobs that are waiting to run are moved to the same relative position on the target queue (for example, jobs with a job queue priority 3 are moved after any other priority 3 jobs that are waiting to run on the target queue).
- Jobs that are held remain held and are placed in the same relative position on the target queue (for example, held jobs with job queue priority 3 are moved after any other priority 3 held jobs on the target queue).
- Jobs that are scheduled to run are moved to the target queue and their scheduled times remain unchanged.

*Character-based interface*

**Command:** Change Job (CHGJOB)

**Example:** The following example moves job JOBA to job queue JOBQB.

```
CHGJOB JOB(JOBA) JOBQ(LIBA/JOBQB)
```

### ***Moving a job up in priority within a job queue***

All jobs in a job queue wait in line for processing. As each job in the queue completes, the next job in line begins. The processing order of the jobs in the queue depends upon the job's priority, and the maximum number of jobs that can run at the same time on the subsystem.

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.

*IBM Navigator for i*

To change a job's priority in the job queue, follow these instructions:

1. Expand **Work Management > All Tasks > Job Queues > Active Job Queues or All Job Queues > The job queue in which your job is located.**
2. Right-click the job and click **Properties.**
3. On the Job - Properties window, click the **Job Queue** tab.
4. 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.
5. 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.

*Character-based interface*

**Command:** Change Job (CHGJOB)

**Parameter:** JOBPTY

**Example:** This command changes the scheduling priority for the job PAYROLL to 4. Because only the simple name of the job is specified, there can be only one job named PAYROLL in the system. If there is more than one, the default of DUPJOBPT(\*SELECT) causes a selection panel to be displayed in an interactive job.

```
CHGJOB JOB(PAYROLL) JOBPTY(4)
```



## ***Tips for setting job priorities***

The priorities for jobs that run in batch environments should normally be lower than priorities for jobs in interactive environments. Also, the time slice should be small enough so that a looping program does not dominate processor time and an activity level.

You may want the priority for the system operator's jobs to be higher than priorities of other jobs so that the system operator can effectively respond to system needs.

If you use QCTL as the controlling subsystem, the operator is automatically running at a higher priority after signing on at the console. This is because QCTL routes the console job using the QCTL class, which specifies a higher priority.

Another way that you can set up your system so that the operator can run at a higher priority would be to use the following instructions:

1. Add a routing entry to the subsystem with unique routing data and specify the QSYS/QCTL class.
2. Create a new job description for the operator, specifying the same unique routing data that you used in the routing entry.
3. Change the operator's user profile to specify the new job description.
4. Now when the operator signs on to that subsystem, the job will route using the QCTL class, which specifies a higher priority than the class used by normal interactive jobs.

The job run priority is the highest priority at which any thread in the job may run. Each thread may have its own thread priority that is lower than the job priority. The Change Job (CHGJOB) command will change only the job priority. The Change Job (QWTCHGJB) API can be used to change either the job priority or a thread priority.

## ***Submitting a job once***

When you need to run a job once, whether immediately or at a scheduled date and time, use the Submit Job (SBMJOB) command. This method places the job on the job queue immediately.

To submit a batch job once, use the character-based interface.

**Command:** Submit Job (SBMJOB)

The SBJJOB command submits a job to a batch job queue by specifying a job description and by specifying a CL command or request data, or specifying routing data to run a program. If you want to run a single CL command in a the batch job, use the CMD parameter on SBJJOB, which does syntax checking and allows prompting.

**Example:** In the following example, the SBJJOB command submits a job named WSYS, using the job description QBATCH, to the job queue QBATCH. The CMD parameter gives the CL command that will run in the job.

```
SBMJOB JOB(QBATCH) JOB(WSYS) JOBQ(QBATCH) CMD(WRKSYSSTS)
```

## **Related concepts**

### The submit job command

This character-based interface command controls the time a job is released in the job queue. It is an easy way to schedule a job that only needs to run once. It allows you to use many of the job attributes defined for your current job.

## ***Viewing job affinity information***

Each job on the system contains memory and processor affinity information.

The affinity information describes whether threads have affinity to the same group of processors and memory as the initial thread when they are started. It also specifies the degree to which the system tries to maintain the affinity between threads and the subset of system resources they are assigned to. In addition, the affinity information specifies whether a job is grouped with other jobs so they have affinity to the same subset of system resources.

By grouping threads together that share a common set of data in main storage, your system's caching and memory access rates can improve.

*IBM Navigator for i*

1. Expand **Work Management > Active Jobs**.

**Note:** You can view the affinity information of a job from any location within work management where you can view jobs.

2. Right-click the job that you want to view, and click **Properties**.

3. On the Resources page, you can view the **Memory and processor affinity** information.

*Character-based interface*

**Command:** Work with Job (WRKJOB)

Select option 3: Display job run attributes, if active

## Managing job descriptions

Since a job description collects a specific set of job-related attributes, the same job description can be used by multiple jobs. Thus, if you use a job description, you do not need to specify the same parameters repeatedly for each job. You can create job descriptions to describe batch jobs or interactive jobs. You can also create unique descriptions for each user of the system. Job descriptions are created and managed by using the character-based interface.

### Creating a job description

You can use the character-based interface, the Work With Job Description (WRKJOB) command or the Create Job Description (CRTJOB) command to create job descriptions.

**Command:** Create Job Description (CRTJOB)

**Example:** In this example, a job description is created named INT4 in the user's current library. This job description is for interactive jobs and is used by Department 127. When you sign on, you must type your password. The characters QCMDI are used as routing data that is compared with the routing table of the subsystem where the job is run. All inquiry messages are compared to the entries in the system reply list to determine whether a reply is issued automatically.

```
CRTJOB JOBDBATCH3 USER(*RQD) RTGDTA(QCMDI)
        INQMSGRPY(*SYSRPLY)
        TEXT('Interactive #4 JOB for Department 127')
```

This command creates a job description named BATCH3 in the user's current library. The jobs using this description are placed on the job queue NIGHTQ. The priority for jobs using this description and their spooled output is 4. QCMDI is the routing data that is compared with entries in the routing table of the subsystem where the job runs. The accounting code of NIGHTQ012345 is used when recording accounting statistics for jobs that use this job description.

```
CRTJOB JOBDBATCH3 USER(*RQD) JOBQ(NIGHTQ) JOBPTY(4)
        OUTPTY(4) ACGCDE(NIGHTQ012345) RTGDTA(QCMDI)
        TEXT('Batch #3 JOB for high priority night work')
```

**Note:** The values in the job description are typically used as the default values of the corresponding parameters in the Batch Job (BCHJOB) and Submit Job (SBMJOB) commands when their parameters are not specified. The values in the job description can be overridden by the values specified on the BCHJOB and SBJOB commands

### Related concepts

#### Job description

The job description allows you to create a set of job attributes that are saved and available for multiple uses. The job description can be used as the source for some of the job attributes that tell the system how to run a job. The attributes tell the system when to start the job, where to get the job from, and how the

job will run. You can think of a job description as a template that many jobs can use, thereby reducing the number of specific parameters that you need to set for each individual job.

### ***Changing a job description***

You can use the character-based interface, the Work With Job Description (WRKJOB) command, or the Change Job Description (CHGJOB) command to change job descriptions.

**Command:** Change Job Description (CHGJOB)

All jobs that are started after the job description is changed that use that job description are affected. If you have changed a job parameter to something different than the what is specified in the job description, that parameter is not affected.

### ***Using a job description***

The most common way to use a job description is by specifying it in the Submit Job (SBMJOB) command. The job description (JOB) parameter is where you specify the job description that you want this job to use. When you define a batch job, you can use the job description in one of two ways:

- Use a specified job description without overriding any of its attributes. For example:

```
SBMJOB JOB(OEDAILY) JOB(QBATCH)
```

- Use a specified job description but override some of the attributes (using the BCHJOB or SBMJOB command). For example, to override the message logging in the job description QBATCH, you specify:

```
SBMJOB JOB(OEDAILY) JOB(QBATCH) LOG(2 20 *SECLVL)
```

The following are additional commands that support the job description parameter:

- Batch Job (BCHJOB): This command indicates the beginning of a batch job in a batch input stream. It can also specify different values for the attributes for the job instead of the ones specified in the job description or user profile for this job. The values contained in the job description or in the user profile named in that job description are used for most parameters not coded in the BCHJOB command.
- Add Prestart Job Entry (ADDPJE): The Add Prestart Job Entry (ADDPJE) command adds a prestart job entry to the specified subsystem description. The entry identifies prestart jobs that may be started when the subsystem is started or when the Start Prestart Jobs (STRPJ) command is entered.
- Add Autostart Job Entry (ADDAJE): The Add Autostart Job Entry (ADDAJE) command adds an autostart job entry to the specified subsystem description. The entry identifies the job name and the job description to be used to automatically start a job.
- Add Work Station Entry (ADDWSE): The Add Work Station Entry (ADDWSE) command adds a workstation entry to the specified subsystem description. Each entry describes one or more workstations that are controlled by the subsystem. The workstations identified in the workstation entries are allowed to sign on or enter the subsystem and run jobs.

**Note:** You cannot override any job description attributes for autostart jobs, workstation jobs, or communications jobs.

### **Related concepts**

#### Job description

The job description allows you to create a set of job attributes that are saved and available for multiple uses. The job description can be used as the source for some of the job attributes that tell the system how to run a job. The attributes tell the system when to start the job, where to get the job from, and how the job will run. You can think of a job description as a template that many jobs can use, thereby reducing the number of specific parameters that you need to set for each individual job.

### ***Controlling the job attribute source***

The attributes that the subsystem assigns to jobs come from five sources; the job description, the user's user profile, a system value, the job issuing the Submit Job (SBMJOB) command, and the workstation (interactive jobs only). You control from where the subsystem retrieves the specific job attribute by

specifying the source in the job description. To modify a job description, use the character-based interface.

**Command:** Change Job Description (CHGJOB)

To control job attributes and tell the subsystem where and when to get job attributes from different system objects, use one of the following:

- \*JOB: Tells the job to get its attributes from the job description.
- \*USRPRF: Tells the job to get its attributes from the user's user profile.
- \*SYSVAL: Tells the job to get its attributes from a system value.
- \*CURRENT: Tells the job to get its attributes from the job issuing the Submit Job (SBMJOB) command.
- \*WRKSTN: Tells the job to get its attributes from the workstation with the job (interactive jobs only).

### ***Deleting a job description***

You can use the character-based interface, the Work With Job Description (WRKJOB) command, or the Delete Job Description (DLTJOB) command to delete job descriptions.

**Command:** Delete Job Description (DLTJOB)

**Note:** Jobs that are already in progress are not affected by this command.

## **Managing batch jobs**

Jobs that do not require user interaction to run can be processed as batch jobs. A batch job typically is a low priority job and can require a special system environment in which to run.

### ***Submitting a batch job***

Since batch jobs are typically low priority jobs that require a special system environment in which to run (such as running at night) they are placed in batch job queues. In the job queue the batch job receives a run time schedule and a priority. To submit a job to a batch job queue, you use the character-based interface and one of two commands.

**Command:** Submit Job (SBMJOB)

**Command:** Submit Database Job (SBMDBJOB)

The difference in these commands is the source of the job:

- The SBMJOB command submits a job to a batch job queue by specifying a job description and by specifying a CL command or request data, or specifying routing data to run a program. If you want to run a single CL command in a the batch job, use the CMD parameter on SBMJOB, which does syntax checking and allows prompting.
- The SBMDBJOB command can be used to submit a job to a batch job queue from a database file. For these jobs, the job description comes from the BCHJOB statement in the input stream.

**Example:** In the following example, the SBMJOB command submits a job named WSYS, using the job description QBATCH, to the job queue QBATCH. The CMD parameter gives the CL command that will run in the job.

```
SBMJOB JOB(QBATCH) JOB(WSYS) JOBQ(QBATCH) CMD(WRKSYSSTS)
```

**Note:** If you get a message that the job was not submitted, you can display the job log spooled file to find errors. Use the WRKJOB command. Specify the job that was not scheduled, select option 4 for spooled files. Display the job log spooled file to find the errors.

### **Related concepts**

[How a batch job starts](#)

When a user submits a batch job, the job gathers information from several system objects before it is placed on a job queue.

[The submit job command](#)

This character-based interface command controls the time a job is released in the job queue. It is an easy way to schedule a job that only needs to run once. It allows you to use many of the job attributes defined for your current job.

## Related information

### [QPRTJOB job](#)

#### *Inline data files*

An inline data file is a data file that is included as part of a batch job when the job is read by a reader or a submit jobs command. You use SBMDBJOB or STRDBRDR to queue a CL batch stream (stream of CL commands to be run). That CL batch stream can include data to be placed into inline data files (temporary files). When the job ends, the inline data files are deleted.

An inline data file is delimited in the job by a `//DATA` command at the start of the file and by an end-of-data delimiter at the end of the file.

The end-of-data delimiter can be a user-defined character string or the default of `//`. The `//` must appear in positions 1 and 2. If your data contains `//` in positions 1 and 2, you should use a unique set of characters, such as `// *** END OF DATA`. To specify this as a unique end-of-data delimiter, the `ENDCHAR` parameter on the `//DATA` command should be coded as:

```
ENDCHAR('// *** END OF DATA')
```

**Note:** Inline data files can be accessed only during the first routing step of a batch job. If a batch job contains a Transfer Job (TFRJOB), a Reroute Job (RRTJOB), or a Transfer Batch Job (TFRBCHJOB) command, the inline data files cannot be accessed in the new routing step.

An inline data file can be either named or unnamed. For an unnamed inline data file, either `QINLINE` is specified as the file name in the `//DATA` command or no name is specified. For a named inline data file, a file name is specified.

A *named inline data file* has the following characteristics:

- It has a unique name in a job. No other inline data file can have the same name.
- It can be used more than once in a job.
- Each time it is opened, it is positioned to the first record.

To use a named inline data file, you must either specify the file name in the program or use an override command to change the file name specified in the program to the name of the inline data file. The file must be opened for input only.

An *unnamed inline data file* has the following characteristics:

- Its name is `QINLINE`. (In a batch job, all unnamed inline data files are given the same name.)
- It can only be used once in a job.
- When more than one unnamed inline data file is included in a job, the files must be in the input stream in the same order as when the files are opened.

To use an unnamed inline data file, do one of the following:

- Specify `QINLINE` in the program.
- Use an override file command to change the file name that is specified in the program to `QINLINE`.

If your high-level language requires unique file names within one program, you can use `QINLINE` as a file name only once. If you need to use more than one unnamed inline data file, you can use an override file command in the program to specify `QINLINE` for additional unnamed inline data files.

**Note:** If you run commands conditionally and process more than one unnamed inline data file, the results cannot be predicted if the wrong unnamed inline data file is used.

### *Considerations for opening inline data files*

You need to consider these elements when you open inline data files.

- The record length specifies the length of the input records. (The record length is optional.) When the record length exceeds the length of the data, a message is sent to your program. The data is padded with blanks. When the record length is less than the data length, the records are truncated.
- When a file is specified in a program, the system searches for the file as a named inline data file before it searches for the file in a library. Therefore, if a named inline data file has the same name as a file that is not an inline data file, the inline data file is always used, even if the file name is qualified by a library name.
- Named inline data files can be shared between programs in the same job by specifying SHARE(\*YES) on a create file or override file command. For example, if an override file command specifying a file named INPUT and SHARE(\*YES) is in a batch job with an inline data file named INPUT, any programs running in the job that specify the file name INPUT shares the same named inline data file. Unnamed inline data files cannot be shared between programs in the same job.
- When you use inline data files, make sure the correct file type is specified on the //DATA command. For example, if the file is to be used as a source file, the file type on the //DATA command must be source.
- Inline data files must be opened for input only.

### ***Starting a batch job that is waiting in the job queue***

Occasionally you might need to force a job to start immediately. While moving the job to a job queue that is not busy is the most efficient method to accomplish this, there are some other methods that you can use.

To start a batch job, first check the status of the job queue in which the job resides and determine if moving the job to another queue makes the most sense for your situation. (**Work Management > All Tasks > Job Queues > Active Job Queues or All Job Queues**)

If moving the job to another queue isn't feasible, you can place the running jobs on hold and then move the job that you need to start up in priority. However, use caution when using this method because the held jobs are still included in the maximum active job count.

To change the priority of the job and indicate when it should run, use the following instructions:

1. Right-click the job and click **Properties**.
2. On the Job Properties window, click the **Job Queue** tab.
3. Change the **Priority on job queue** to a higher priority (0 is the highest).
4. Set the **When to make job available to run** to either Now or specify the date and time.
5. Click **OK**.

### **Related concepts**

#### How a batch job starts

When a user submits a batch job, the job gathers information from several system objects before it is placed on a job queue.

### **Related information**

#### QPRTJOB job

## **Managing interactive jobs**

An interactive job starts when you sign on to the system or, transfer to a secondary or group job. The interactive job ends when you sign off. Working from a display station, you interact with the system by issuing commands, using function keys, and running programs and applications. The following information discusses the various methods for managing and controlling interactive jobs.

### ***Controlling inactive jobs and workstations***

You can control the amount of time the workstation can remain inactive before the subsystem sends a message (called time-out) by specifying a time interval in the Time-out interval for inactive jobs

(QINACTIV) system value. Controlling inactive jobs provides security so that users do not leave signed on displays inactive.

## How the system determines a workstation is inactive

The subsystem determines that a workstation is inactive if all of the following are true:

- The job has not processed any additional transactions during the timer interval.

**Note:** A transaction is defined as any operator interaction, like scrolling, pressing enter, pressing function keys, and so on. Typing at the workstation without pressing enter is not considered a transaction. If a job at the workstation does not meet the inactive criteria, the job is considered active.

- The job status is display wait.
- The job is not disconnected.
- The job status has not changed.
- The subsystem in which the job is running is not in the restricted state.

## Handling inactive jobs

To handle an inactive job found on the system, use the When a job reaches time-out (QINACTMSGQ) system value. To determine the processing options choose from the following:

- Set the QINACTMSGQ system value to a message queue name.

If you specify a message queue name for the QINACTMSGQ system value, a user or program can monitor the message queue and take whatever action is needed, such as ending a job.

If a workstation with a secondary job pair is inactive, the system sends two messages (one of each of the secondary job pairs) to the message queue. The user or program can then use either the ENDJOB command against one or both secondary jobs, or the DSCJOB command against the active job at the display.

- Set the QINACTMSGQ system value to \*DSCJOB.

If you specify \*DSCJOB for the QINACTMSGQ system value, the system disconnects all jobs at the workstation. The system sends a message that indicates that all jobs at the workstation have disconnected from QSYSOPR or the configured message queue. (A configured message queue is the message queue specified in the MSGQ parameter of the display device description. By default it is QSYS or QSYSOPR.) If the interactive job does not support disconnecting the job, the job ends instead.

A message continues to be sent for each interval that the job is inactive.

- Set the system value QINACTMSGQ to \*ENDJOB.

If you specify \*ENDJOB for the QINACTMSGQ system value, the system ends all of the jobs at the workstation. The system sends a message that indicates that all jobs at the workstation have ended to QSYSOPR or the configured message queue.

**Note:** Source pass-through jobs, client VTM (virtual terminal manager) jobs, and 3270 device emulation jobs are excluded from the time-out because they always appear inactive. System/36 environment MRT jobs are also excluded since they appear as batch jobs.

## Ending interactive jobs

You can use several different methods to end an interactive job.

You can use IBM Navigator for i to end the job.

1. From the Confirm Delete/End window, you can specify whether you want this interactive job to end in a controlled manner or immediately.
2. You can use the End Job (ENDJOB) character-based interface command.



3. To end an interactive job immediately using the character-based interface, use the Sign Off (SIGNOFF) command at the workstation. To end the connection through the network, use the end connection parameter (ENDCNN) on the SIGNOFF command.
4. To disconnect all jobs from a device, use the Disconnect Job (DSCJOB) command.

To use IBM Navigator for i and the Confirm Delete/End window, use the following instructions:

1. Expand **Work Management > Active Jobs**.
2. Right-click the job that you want to end and click **Delete/End**.

The Confirm Delete/End window appears where you can specify how and when you want the interactive job to end.

**Note:** To end all of the interactive jobs associated with the workstation, or all jobs associated with the group (if the job is a group job), set the value of the **Action for related interactive jobs** field to either End for group jobs or End all (this is equivalent to the ADLINTJOBS parameter on the ENDJOB command).

You can also request the subsystem to send a message to a message queue when an interactive job has been inactive for a specified period of time. You, or a program monitoring that message queue, can then end or disconnect the job.

### **Related concepts**

#### Disconnecting interactive jobs

When the Disconnect Job (DSCJOB) command is called, the job is disconnected and the sign-on display is shown again. To connect with the job again, sign on to the same device from which you disconnected. Another interactive job may be started on the device under a different user name.

### ***Disconnecting all jobs from a device***

The Disconnect Job (DSCJOB) command allows the interactive user to disconnect all interactive jobs at the workstation and return to the sign-on display. The switched line is dropped only if that is specified in the workstation device description of this workstation and if no other workstation on this line is active. If the job is disconnected when the disconnect interval in the Time-out interval for disconnected jobs (QDSCJOBITV) system value is reached, the job is ended and the job log is not included with the job's spooled output.

Restrictions:

1. A job being disconnected must be an interactive job.
2. A job which is being held cannot be disconnected.
3. A pass-through job cannot be disconnected unless the user has used the system request function to return to the source system from the pass-through target system.
4. The command must be issued from within the job being disconnected, or the issuer of the command must be running under a user profile which is the same as the job user identity of the job being disconnected, or the issuer of the command must be running under a user profile which has job control (\*JOBCTL) special authority.
5. The job user identity is the name of the user profile by which a job is known to other jobs.
6. A job cannot be disconnected if PC organizer is active.

**Command:** Disconnect Job (DSCJOB)

### **Related concepts**

#### Disconnecting interactive jobs

When the Disconnect Job (DSCJOB) command is called, the job is disconnected and the sign-on display is shown again. To connect with the job again, sign on to the same device from which you disconnected. Another interactive job may be started on the device under a different user name.

### ***Job disconnection considerations***

There are several factors that you must consider whenever you disconnect a job.

- An option on the System Request menu allows you to disconnect an interactive job, causing the sign-on display to appear. The option calls the Disconnect Job DSCJOB command.



- When connecting with a job again, the values specified on the sign-on display for program, menu, and current library are ignored.
- A job which has PC organizer or PC text assist function active cannot be disconnected.
- If the job cannot be disconnected for any reason, the job will be ended instead.
- All disconnected jobs in the subsystem end when the subsystem ends. If a subsystem is ending, the DSCJOB command cannot be issued in any of the jobs in the subsystem.
- The system value Disconnect Job Interval (QDSCJOBITV) can be used to indicate a time interval for which a job can be disconnected. If the time interval is reached, the disconnected job ends
- Disconnected jobs that have not exceeded the QDSCJOBITV system value will end when the subsystem is ended or when an IPL occurs.

### **Related concepts**

#### Disconnecting interactive jobs

When the Disconnect Job (DSCJOB) command is called, the job is disconnected and the sign-on display is shown again. To connect with the job again, sign on to the same device from which you disconnected. Another interactive job may be started on the device under a different user name.

### ***Avoiding a long-running function from a workstation***

To avoid a long-running function (such as save/restore) from a workstation without tying it up, the system operator can submit the job to a job queue.

The subsystem description QSYS/QBATC or QSYS/QBASE, which is supplied by IBM, has a job queue QSYS/QBATC that can be used for this purpose. If you created your own subsystem, you should refer to the job queue for that subsystem. The system operator can submit the commands from the system operator menu.

The following is an example of submitting a long-running command:

```
SBMJOB JOB(SAVELIB) JOB(QBATC) JOB(QSYS/QBATC)
      CMD(SAVLIB LIBX DEV(DKT01))
```

### **Related concepts**

#### How an interactive job starts

When a user signs on to the system, the subsystem gathers information from several system objects before the interactive job is ready.

## **Managing prestart jobs**

You can use prestart jobs to reduce the amount of time required to handle a program start request. These are the most common tasks associated with prestart jobs that you can perform.

### **Related concepts**

#### Prestart communications jobs and job accounting

If your system uses job accounting, the prestart job program should run the Change Prestart Job (CHGPJ) command with the program start request value for the accounting code parameter (CHGPJ ACGCDE(\*PGMSTRRQS)) immediately after the program start request attaches to the prestart job.

### ***Starting a prestart job***

Prestart jobs typically start at the same time the subsystem is started. You manually start a prestart job when all prestart jobs have been ended by the system due to an error or were never started during subsystem start up due to STRJOBS (\*NO) on the prestart job entry. To start a prestart job, use the character-based interface.

**Command:** Start Prestart Jobs (STRPJ )

The STRPJ command should not be used until the startup of the related subsystem is complete. To make sure that the necessary prestart job successfully starts, code a delay loop with a retry if the STRPJ command fails.

The number of prestart jobs that can be active at the same time is limited by the MAXJOBS attribute on the prestart job entry and by the MAXJOBS attribute for the subsystem. The MAXACT attribute on the communications entry controls the number of program start requests that can be serviced through the communications entry at the same time.

**Note:** If you specified \*NO on the STRJOBS attribute, no prestart jobs start for the prestart job entry when the subsystem starts. Running the STRPJ command does not cause the value of the STRJOBS parameter to change.

**Example:** This example starts prestart jobs for prestart job entry PJPGM in subsystem SBS1. Subsystem SBS1 must be active when this command is issued. The number of jobs started is the number specified in the INLJOBS value of prestart job entry PJPGM. The subsystem starts program PJPGM in library PJLIB.

```
STRPJ  SBS(SBS1)  PGM(PJLIB/PJPGM)
```

## Related concepts

### Prestart jobs

A prestart job is a batch job that starts running before a work request is received. The prestart jobs are started before any other types of jobs in a subsystem. Prestart jobs are different from other jobs because they use prestart job entries (part of the subsystem description) to determine which program, class, and storage pool to use when they are started.

## Related information

Experience Report: Tuning prestart job entries

## ***Queueing or rejecting program start requests***

If a program start request arrives when the current number of prestart jobs is less than the number specified in the MAXJOBS attribute on the prestart job entry, and none of the prestart jobs are available to handle the program start request, you have the option to have this new request rejected or queued.

To reject or queue the program start request, use the WAIT attribute on the prestart job entry.

WAIT(\*NO) means that if no prestart job is available immediately, the program start request is rejected.

WAIT (\*YES) means that if no prestart job is available immediately and no prestart job can be started due to MAXJOBS to service the program start request, the program start request is rejected. If no prestart job is available immediately, but additional prestart jobs can be or have been started, the program start request is queued.

This command adds a prestart job entry for the PGM1 program in the QGPL library to the PJSBS subsystem description contained in the QGPL library. The entry specifies that 15 prestart jobs (program PGM1 in the QGPL library) are started when subsystem PJSBS in the QGPL library is started. When the pool of available prestart jobs is reduced to four (because the prestart jobs are servicing requests specified for program PGM1 in the QGPL library), ten additional jobs are started. If no prestart jobs are available for this entry when a request is received, the request is rejected.

```
ADDPJE  SBSD(QGPL/PJSBS)  PGM(QGPL/PGM1)  INLJOBS(15)  
        THRESHOLD(5)  ADLJOBS(10)  WAIT(*NO)
```

## ***Tuning prestart job entries***

You should have enough prestart jobs started by the subsystem so that work is handled as it arrives rather than waiting for new jobs to be started. These tips show how to tune your prestart jobs for optimum performance.

## Related tasks

### Configuring a server subsystem

The information in this section explains how to set up a subsystem for server jobs.

### *Setting the number of prestart jobs*

While the system is handling its normal workload and information about the workload is available, follow the following steps:

1. Use the Work with Subsystems (WRKSBS) command to get a list of all active subsystems. For each subsystem in the list of active subsystems, use option 5 to display the subsystem description.

On the Display Subsystem Description panel, use option 10 to display prestart job entries. If there are no prestart job entries for that subsystem description, continue with the next subsystem in the WRKSBS list.

2. On the Display Prestart Job Entries panel, use option 5 to display details for the prestart job entry. Make a note of the current settings for Initial number of jobs, Threshold, and Additional number of jobs.
3. For each prestart job entry in the subsystem description, enter a Display Active Prestart Jobs (DSPACTPJ) command.

For example:

```
DSPACTPJ SBS(SUBSYSTEM) PGM(PJPGMLIB/PJPROGRAM)
```

If the DSPACTPJ command is not currently allowed, the prestart job entry is not active and does not need to be changed. Continue with the next prestart job entry or the next subsystem description.

4. Use the DSPACTPJ information to get an estimate of your workload. The DSPACTPJ command produces a display that looks like this:

```
-----
                                Display Active Prestart Jobs                                SYSTEM
                                                                                   08/06/03 07:35:00
Subsystem . . . . . : SUBSYSTEM      Reset date . . . . . : 08/06/03
Program . . . . . : PJPROGRAM       Reset time . . . . . : 07:23:03
Library . . . . . : PJPGMLIB       Elapsed time . . . . . : 0000:11:57

Prestart jobs:
Current number . . . . . : 122
Average number . . . . . : 21.4
Peak number . . . . . : 122

Prestart jobs in use:
Current number . . . . . : 120
Average number . . . . . : 17.7
Peak number . . . . . : 120

More...

Press Enter to continue.

F3=Exit  F5=Refresh  F12=Cancel  F13=Reset statistics
-----
```

```
-----
                                Display Active Prestart Jobs                                SYSTEM
                                                                                   08/06/03 07:35:00
Subsystem . . . . . : SUBSYSTEM      Reset date . . . . . : 08/06/03
Program . . . . . : PJPROGRAM       Reset time . . . . . : 07:23:03
Library . . . . . : PJPGMLIB       Elapsed time . . . . . : 0000:11:57

Program start requests:
Current number waiting . . . . . : 0
Average number waiting . . . . . : .0
Peak number waiting . . . . . : 1
Average wait time . . . . . : 00:00:00.0
Number accepted . . . . . : 120
Number rejected . . . . . : 0

Bottom

Press Enter to continue.

F3=Exit  F5=Refresh  F12=Cancel  F13=Reset statistics
-----
```

Find the prestart jobs in use section and the value for the peak number. In this example, the value is 120. This number is an estimate of your peak workload. Make a note of this value, it is used in the following steps.

Find the program start requests section and the value for the peak number waiting. You may need to page down to see this field. In this example, the value is 1. This number tells you how well the system is handling the arrival of new work. Make a note of this value, it is used in the following steps.

5. If DSPACTPJ shows a zero (0) for the peak number of prestart jobs in use, the prestart job entry is not being used by your workload and therefore does not need to be changed. Continue with the next prestart job entry or the next subsystem description.
6. Choose a value for the THRESHOLD parameter. When the pool of available jobs is reduced below this number, more jobs are started. Starting jobs takes time. Meanwhile, more requests for work may arrive. Set THRESHOLD to a value of at least one plus the number of requests that can arrive while new jobs are being started

In this example, the value chosen is 10. This is an estimate of arrival of work requests, a guess based on the peak number of jobs in use. This is not an accurate analysis of hard-to-get measurements.

Refer to the notes you took in an earlier step. If the current setting for THRESHOLD is high enough, the peak number waiting is zero. If the peak number waiting is not zero, add this number to your current THRESHOLD value and compare the result to the estimated value based on arrivals. Use the larger value. The sample DSPACTPJ information shows a value of 1 which means the current value for THRESHOLD is too low. The current setting plus one is less than the estimate of 10. For this example, we use the value 10.

7. Choose a value for the initial number of jobs (INLJOBS) parameter. INLJOBS specifies the number of jobs that are started when the subsystem is started. Also, INLJOBS is part of what the subsystem uses to decide if there are too many prestart jobs waiting for work.

Refer to the notes you took in an earlier step. Use the peak number of prestart jobs in use as the estimate for peak workload, add the value for THRESHOLD, and use the result as the new value for INLJOBS. The DSPACTPJ information shows a peak of 120 prestart jobs in use and we have already chosen a THRESHOLD of 10, so the new value chosen for INLJOBS is 130.

8. Choose a value for the additional number of jobs (ADLJOBS) parameter. ADLJOBS specifies the additional number of prestart jobs that are started when the number of available prestart jobs drops below the value specified on the Threshold (THRESHOLD) parameter.

When INLJOBS and THRESHOLD are high enough to avoid causing requests to wait, ADLJOBS can be fairly low. If INLJOBS is far below peak workload, ADLJOBS may need to be as high as THRESHOLD. In this example, the chosen value is 5.

Try to avoid large numbers. If you specify a large value for ADLJOBS, the subsystem starts a large number of jobs all at once. This can adversely affect system performance and it delays the subsystem's handling of other work.

9. Compare the newly chosen values with the values configured in the prestart job entry. To be sure to have enough prestart jobs, use the larger value for each parameter. Change the configured values by using the Change Prestart Job Entry (CHGPJE) command.

```
CHGPJE SBSDB(SBSLIB/SUBSYSTEM) PGM(PJPGMLIB/PJPROGRAM)
      INLJOBS(130) THRESHOLD(10) ADLJOBS(5)
```

10. Continue with the next prestart job entry or the next subsystem description.

## Details

Some additional details may help you make good decisions when following this procedure.

- If the THRESHOLD value is too small, work waits for new jobs to be started. In some cases, errors occur because requests time out.

Consider an example where THRESHOLD is 2 and there are only two jobs waiting for work. When the next work request arrives, that request is given to one of the waiting jobs and additional jobs are started. In this example, two more requests arrive before the new jobs are ready. The first request is handled by a waiting job. The second request waits for one of the new jobs to become ready. For the example workload, THRESHOLD should be set to at least 3: one to trigger the creation of more jobs plus two for the number of requests that arrive while new jobs are being started.

- Because the subsystem starts jobs when they are needed, the subsystem also ends jobs when they are not needed. This happens for prestart job entries that specify a maximum number of uses (MAXUSE) greater than one. The value for the INLJOBS parameter tells the subsystem how many jobs are needed. You need to get INLJOBS set correctly to prevent the subsystem from ending too many jobs.

If the INLJOBS value is too small, the subsystem periodically starts jobs because there are too few and end jobs because there are too many. Moreover, the system incurs the cost of starting new jobs at the time when the system is most busy.

- In the sample output from the DSPACTPJ command, the peak number of prestart jobs in use is 120 while the average number of prestart jobs in use is 17.7. This is not a high peak. This is a low average. By default, DSPACTPJ shows what has happened since the subsystem started. The average includes periods when the workload is zero.

Even when you use F13 to reset the statistics and even when you carefully control the sample interval, the average number of prestart jobs in use is likely to be lower than the number you should tune to. A workload may have an average somewhere between 40 and 60 jobs and yet have lots of peaks between 100 and 120 jobs.

When INLJOBS is set to the estimated peak workload plus THRESHOLD, the subsystem does not need to start additional jobs unless the actual workload exceeds the estimated peak workload. If your workload has peaks that are relatively high and relatively infrequent, you may wish to set INLJOBS to a lower number.

- The procedure given in this topic assumes that the peak load on a typical day is a typical peak load. If you gather more data, you might be able to produce a better estimate of your workload.

You can use the List Job (QUSLJOB) API or the Open List of Jobs (QGYOLJOB) API to periodically sample your workload. For some workloads, it helps to graph the results. You do not need a perfect prediction for the number of prestart jobs. You only need to be close enough to avoid delays and time outs.

- If THRESHOLD and INLJOBS are too large, there are active jobs in the subsystem that are not needed. Starting and ending extra jobs takes more time when starting or ending the subsystem or when starting or ending the prestart job entry.

It is better to use values that are slightly higher than what is needed than to use values that are lower than what is needed. Having a few extra jobs is not a problem because these jobs are waiting for work and do not compete for memory or processors.

- Because prestart jobs were first used with communications devices, a request for work is called a program start request and the prestart job shows a status of PSRW (waiting for program start request) when it is waiting for work.

### ***Changing job attributes for prestart jobs***

Large job message queues can consume storage, can lead to large job logs which also consume storage, and can cause IPL performance problems when job message queues need recovery or cleanup during an IPL. This example shows how to change the job message queue full action (JOBMSGQFL) and job message queue maximum size (JOBMSGQMX) values for prestart jobs.

**Note:** The QDFTSVR job description was introduced in release V5R3M0 to do some of this for you.

To limit the size of job message queues for prestart jobs without affecting other jobs, follow these steps:

1. Find the prestart jobs that you want to affect and determine which job description is used by the prestart job entry. (To do this, use the Display Subsystem Description (DSPSBSD) command.)
2. Determine whether the job description is used by just the one prestart job entry (in which case you can just modify that job description) or used by multiple references such as user profiles, prestart job entries, other SBSD entries, and so on. (You can always create another job description for the "don't know" but if you know that a change to the existing job description affects only the jobs you want affected then you should just modify that particular job description.)
3. Create a new job description to be used by the prestart job entries that you want to affect. You can use the Create Job Description (CRTJOBBD) command, but in this example we make a copy of the job description that is currently being used.

**Note:** If you have the job description JOBD(\*USRPRF) you can use the Display User Profile (DSPUSRPRF) command to determine which job description is currently being used. The default configurations use job description QDFTJOB or QDFTSVR.

```
DSPUSRPRF USRPRF(QUSER)
```

To avoid confusion with IBM-supplied objects, avoid names starting with the letter 'Q'. This example uses the name PJJOB as the name of the job description for the prestart job entries. Use the Create Duplicate Object (CRTDUPOBJ) command to make a copy of the job description currently used by the QUSER user profile.

```
CRTDUPOBJ OBJ(QDFTSVR) FROMLIB(QGPL) OBJTYPE(*JOB)  
          TOLIB(QGPL) NEWOBJ(PJJOB)
```

- Match the object ownership and authorities of the job description that you copied. Since QDFTSVR and QDFTJOB are owned by QPGMR, the example (below) shows how to change the newly created job description to be owned by QPGMR. Use the Change Object Owner (CHGOBJOWN) command and the Grant Object Authority (GRTOBJAUT) command to get the object ownership and public authority set correctly. You can find the owner and authorities by using the Display Object Authority (DSPOBJAUT) command.

```
CHGOBJOWN OBJ(QGPL/PJJOB) OBJTYPE(*JOB) NEWOWN(QPGMR)  
GRTOBJAUT OBJ(QGPL/PJJOB) OBJTYPE(*JOB) USER(*PUBLIC) AUT(*USE)
```

- Use the Change Job Description (CHGJOB) command to customize the job attributes. In this example, we use a value of 8 megabytes for the job message queue maximum size. Other values would also work as long as the limit is far less than 64 megabytes.

```
CHGJOB JOB(QGPL/PJJOB) JOBMSGQMX(8) JOBMSGQFL(*WRAP)  
      TEXT('Job attributes for prestart job entries')
```

- Look through all of the prestart job entries that are active on your system. Use the Work with Subsystems (WRKSBS) command to get a list of all active subsystems. Use option 5 to display the subsystem description. Use option 10 to display prestart job entries and use option 5 to display details for the prestart job entry.

If the prestart job entry specifies USER(QUSER) and JOBD(\*USRPRF), use the Change Prestart Job Entry (CHGPJE) command to specify the new job description.

```
CHGPJE SBSL(SBSLIB/SUBSYSTEM) PGM(PJPGMLIB/PJPROGRAM)  
      JOBD(QGPL/PJJOB)
```

If the prestart job entry specifies a job description, use the Change Job Description (CHGJOB) command to change the JOBMSGQMX and JOBMSGQFL values in that job description.

```
CHGJOB JOB(JOBDLIB/JOBDNAME) JOBMSGQMX(8) JOBMSGQFL(*WRAP)
```

## Details

The QDFTJOB job description is used by many prestart job entries and is used many other places in the system. This example creates a single new job description named PJJOB. The new job description is used by many prestart job entries but it is not used elsewhere. To use different values for different prestart job entries, use a different job description for each entry. Some prestart job entries already have unique job descriptions.

Some job attributes for prestart jobs cannot be changed using this procedure because they do not come from the job description that is used when starting the job. Many servers that use prestart jobs swap user profiles and then use the Change Job (QWTCCHJB) API to change a subset of the job attributes. The changed job attributes come from the job description used by the user profile to which the prestart job has swapped. Refer to the JOBC0300 format of the Change Job API for more information.

For some job attributes, the job description may indicate that the value is to be taken from a system value. When you change the system value, the change affects all jobs that get their job attribute from the system

value. Changing the value in the job description affects only those jobs that get their job attributes from that job description.

## Ending a prestart job

You can use the character-based interface to end a prestart job in an active subsystem.

Jobs can be waiting for a request or can already be associated with a request. Spooled output files associated with the jobs being ended can also be ended or allowed to remain on the output queue. The limit on the number of messages being written to each of the job logs can also be changed.

**Note:** To end all of the jobs for a prestart job entry, in an active subsystem, use the End Prestart Job (ENDPJ) command. If however, you intend to only end a specific prestart job that is in trouble, use the End Job (ENDJOB) command against the specific prestart job.

**Command :** End Prestart Job (ENDPJ)

**Example:** This command ends all jobs associated with prestart job entry PJPGM in subsystem SBS1 immediately. Spooled output produced by these prestart jobs is deleted and the job log is saved.

```
ENDPJ  SBS(SBS1)  PGM(PJLIB/PJPGM)  OPTION(*IMMED)
        SPLFILE(*YES)
```

**Example:** This command ends all the jobs associated with prestart job entry PJPGM2 in subsystem SBS2. Spooled output for these prestart jobs is saved for normal processing by the spooling writer. The jobs have 50 seconds to perform any cleanup routines, after which they are immediately ended.

```
ENDPJ  SBS(SBS2)  PGM(PJPGM2)  OPTION(*CNTRL)
        DELAY(50)  SPLFILE(NO)
```

## Related concepts

### Prestart jobs

A prestart job is a batch job that starts running before a work request is received. The prestart jobs are started before any other types of jobs in a subsystem. Prestart jobs are different from other jobs because they use prestart job entries (part of the subsystem description) to determine which program, class, and storage pool to use when they are started.

## Related information

[Experience Report: Tuning prestart job entries](#)

## Managing job class objects

A class object contains the run attributes that control the run-time environment of a job. IBM-supplied class objects, or classes, meet the needs of both typical interactive and batch applications. The class used by a job is specified in the subsystem description routing entry used to start the job. If a job consists of multiple routing steps, the class used by each subsequent routing step is specified in the routing entry used to start the routing step.

## Creating a class object

You can create a class object by using the character-based interface. The class defines the processing attributes for jobs that use the class. The class used by a job is specified in the subsystem description routing entry used to start the job. If a job consists of multiple routing steps, the class used by each subsequent routing step is specified in the routing entry used to start the routing step.

**Command:** Create Class (CRTCLS)

**Example:** This example creates a class called CLASS1. The class is stored in the current library specified for the job. The user text 'This class for all batch jobs from Dept 4836' describes the class. The attributes of this class provide a run priority of 60 and a time slice of 900 milliseconds. If the job has not finished running at the end of a time slice, it is eligible to be moved out of main storage until it is allocated another time slice. The defaults for the other parameters are assumed.

```
CRTCLS  CLS(CLASS1)  RUNPTY(60)  TIMESLICE(900)
        TEXT('This class for all batch jobs from Dept 4836')
```

## Related concepts

### Class object

A class object contains the run attributes that control the run-time environment of a job. IBM-supplied class objects, or classes, meet the needs of both typical interactive and batch applications. The following classes (by name) are supplied with the system:

### ***Changing a class object***

You can change the attributes of a class object by using the character-based interface. Any attribute can be changed, except for the public authority attribute. Refer to the Revoke Object Authority (RVKOBJAUT) command and the Grant Object Authority (GRTOBJAUT) command for more information about changing object authorizations.

**Command:** Change Class (CHGCLS)

**Example:** This command changes a class called CLASS1 in the library on the job's library list. The run priority for the class is changed to 60 and a time slice of 900 milliseconds.

```
CHGCLS  CLS(CLASS1)  RUNPTY(60)  TIMESLICE(900)
```

## Related concepts

### Class object

A class object contains the run attributes that control the run-time environment of a job. IBM-supplied class objects, or classes, meet the needs of both typical interactive and batch applications. The following classes (by name) are supplied with the system:

## Managing threads

You can perform many tasks when managing threads.

### ***Viewing threads running under a specific job***

Every active job running on your 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.

### **Related concepts**

#### Threads

The term *thread* is shorthand for "thread of control". A thread is the path taken by a program while running, the steps performed, and the order in which the steps are performed. A thread runs code from its starting location in an ordered, predefined sequence for a given set of inputs.

### **Related information**

[Example: End a thread using Java](#)

[Thread management APIs](#)

*IBM Navigator for i*

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

1. Expand **Work Management > Active Jobs**.
2. Right-click the job with which you want to work, and click **Details > Threads**.

*Character-based interface*

**Command:** Work With Job (WRKJOB)

**Example:** The following example displays the Work With Threads screen for the job Crtpfrdta.

```
WRKJOB  JOB(Crtpfrdta)  OPTION(*THREAD)
```



## ***What you can do with threads***

Since 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. This helps you to keep the job running efficiently. You can use IBM Navigator for i to find the thread you want to manage.

After you have located the thread, you can right-click the thread and select one of the following actions:

### **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 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.

### **Thread Properties**

Displays the different attributes of a thread.

For more detailed information about the actions you can perform on threads, see the IBM Navigator for i online help.

### **Related information**

[Performance system values: Thread affinity](#)

[Performance system values: Automatically adjust thread resources](#)

## ***Viewing 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.

### **Related concepts**

#### Threads

The term *thread* is shorthand for "thread of control". A thread is the path taken by a program while running, the steps performed, and the order in which the steps are performed. A thread runs code from its starting location in an ordered, predefined sequence for a given set of inputs.

### **Related information**

[Example: End a thread using Java](#)

[Thread management APIs](#)

### *IBM Navigator for i*

To view the attributes of a thread, use the following instructions:

1. Expand **Work Management** > **Active Jobs** or **Server Jobs**.
2. Right-click the job with which you want to work, and click **Details** > **Threads**.
3. Right-click the thread with which you want to work, and click **Properties**.

The information under the General tab allows you to view the attributes of a thread. These attributes include the thread identifier, the detailed status of the thread, the current user, the type of thread running, the job the thread is running under, and the disk pool group the thread is running in.

The information under the Performance tab allows you to view basic performance elements and let you change the run priority of the thread. **Run priority** indicates the importance of the thread in relation to other threads running in the system. The possible values range from job priority to 99 (meaning the highest possible priority will vary). The thread run priority may never be higher than the run priority for the job in which the thread is running.

You can view the performance values calculated since the thread started, which include CPU and total disk I/O. You can also view, refresh, set up an automatic refresh, or reset the **Elapsed performance statistics** that have been calculated for a thread.

#### *Character-based interface*

**Command:** Work With Job (WRKJOB)

**Example:** The following example displays the Work With Threads screen for the job Crtpfrdta.

```
WRKJOB JOB(Crtpfrdta) OPTION(*THREAD)
```

### **Ending or deleting threads**

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 might 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 might or might 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 might or might not continue to run. If they continue running without the thread that you end, they might produce undesirable results.

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

#### **Related concepts**

##### Threads

The term *thread* is shorthand for "thread of control". A thread is the path taken by a program while running, the steps performed, and the order in which the steps are performed. A thread runs code from its starting location in an ordered, predefined sequence for a given set of inputs.

#### **Related information**

Example: End a thread using Java

Thread management APIs

#### *IBM Navigator for i*

To delete or end a thread, use the following instructions:

1. Expand **Work Management** > **Active Jobs** or **Server Jobs**.
2. Right-click the job with which you want to work, and click **Details**, and then **Threads**.
3. Right-click the thread with which you want to end, and click **Delete/End**.

#### *Character-based interface*

**Command:** Work With Job (WRKJOB) Option 20: **Work with threads, if active**

**Example:** The following example displays the Work With Threads screen for the job Crtpfrdta.

```
WRKJOB JOB(Crtpfrdta) OPTION(*THREAD)
```

At the Work With Threads screen, select Option: 4=End.

## Managing job scheduling

You can schedule a job to run using the Advanced Job Scheduler, by using the IBM Navigator for i Job Properties window, or by changing the job schedule entry via the character-based interface.

### Scheduling a batch job using IBM Navigator for i

The Job Properties - Job Queue window provides a way for you to schedule a batch job to run now or run once at a specific date and time.

To schedule a job using IBM Navigator for i, use the following instructions:

1. Expand **Work Management** > **Job Queues** > **Active Job Queues or All Job Queues** > **The job queue that contains your job.**
2. Right-click the job and click **Properties.**
3. On the Job Properties window, click the Job Queues tab.
4. To schedule the job, use the options that are located under **When to make job available to run.**

For information about how to use this window, see the IBM Navigator for i help.

### Scheduling a job using Management Central Scheduler

If you do not have the plug-in Advanced Job Scheduler installed, you can use the Management Central Scheduler to schedule jobs.

You can start the Management Central scheduler by clicking on the **Schedule** button that appears on many of the IBM Navigator for i windows. For example suppose you want to use IBM Navigator for i Run Command window to submit a clean up job but you do not want that job to run until after peak hours.

1. From IBM Navigator for i, right-click the server that you want to run the clean up job and click **Run Command.**
2. From the Run Command window, type in the character-based syntax for running your job. If you need assistance type in the first command and click **Prompt.**
3. When you have completed the command, click **Schedule.** The Management Central Scheduler window displays where you can schedule this job to run once, or run as a recurring job.

You can schedule a task to run once, in which case the task runs a single time beginning at the specified date and time. Tasks that run only once are removed from the Scheduled Tasks container when they run. They then appear in a Task Activity container.

**Important:** Do not use the Work with Job Schedule Entries (WRKJOBSCDE) to alter or delete a scheduled job if that job was scheduled using the Management Central Scheduler or the Advanced Job Scheduler. If the job is altered or deleted by using WRKJOBSCDE, Management Central is not notified of the changes. The task might not run as expected, and error messages can appear in the Management Central server job logs.

If it is necessary to make a change to a job that was schedule using the Management Central Scheduler or the Advanced Job Scheduler, use the IBM Navigator for i interface.

#### Related concepts

[Management Central scheduler](#)

System i® Navigator provides an integrated scheduler, the Management Central scheduler, to organize when you want your jobs to process. You have the option of choosing to perform a task immediately or

choosing a later time. You can use the Management Central scheduler to schedule almost any task in Management Central.

## Working with job schedule entries

In addition to the IBM Navigator for i Job Properties - Job Queue window, you can also change the job schedule entry directly by using the character-based interface. The following is a list of common character-based interface tasks that you can use when working with job schedule entries.

**Important:** Do not use the Work with Job Schedule Entries (WRKJOBSCDE) to alter or delete a scheduled job that was scheduled using the Management Central Scheduler or the Advanced Job Scheduler. If the job is altered or deleted by using WRKJOBSCDE, Management Central is not notified of the changes. The task may not run as expected, and error messages can appear in the Management Central server job logs.

### Related concepts

#### Job schedule entries

If your system does not have the Management Central Scheduler or the Advanced Job Scheduler, you can still schedule jobs using a job schedule entry, which is accessed from the character-based interface. Using this method you can schedule jobs to recur or to run only once.

### Adding a job schedule entry

The Add Job Schedule Entry (ADDJOBSCDE) command allows you to schedule batch jobs by adding an entry to the job schedule. You can use this command to schedule a batch job to be submitted once, or to schedule a batch job to be submitted at regular intervals.

**Command:** Add Job Schedule Entry (ADDJOBSCDE)

**Example:** This command submits a job named CLEANUP every Friday at 11 p.m. The job uses job description CLNUPJOB in library CLNUPLIB. If the system is powered down or is in the restricted state at 11 p.m. on Friday, the job is not submitted at IPL or when the system comes out of restricted state.

```
ADDJOBSCDE  JOB(CLEANUP)  SCDDATE(*NONE)
              CMD(CALL PGM(CLNUPLIB/CLNUPPGM))
              SCDDAY(*FRI)   SCDTIME('23:00:00')
              FRQ(*WEEKLY)   RCYACN(*NOSBM)
              JOB(CLNUPLIB/CLNUPJOB)
```

### Changing a job schedule entry

This command changes the entry in the job schedule, but it does not affect any jobs already submitted using this entry. To change a job entry, use the character-based interface.

To change a job schedule entry, you must have the same authorities that are required to add an entry. However, the authorities to the individual objects are checked only if you are changing that parameter for the entry. In addition, if you do not have \*JOBCTL special authority, you can change only the entries that your user profile added to the job schedule object.

**Command:** Change Job Schedule Entry (CHGJOBSCDE)

**Example:** This command changes job schedule entry BACKUP number 001584 so that its jobs are submitted to job queue QBATCH in library QGPL.

```
CHGJOBSCDE  JOB(BACKUP)  ENTRYNBR(001584)  JOBQ(QGPL/QBATCH)
```

**Example:** This command changes the schedule of a batch job to run program A at 11 a.m. on 12/15/03 and every week on that same day.

```
CHGJOBSCDE  JOB(EXAMPLE)  ENTRYNBR(*ONLY)  CMD(CALL PGM(A))
              FRQ(*WEEKLY)  SCDDATE(121503)  SCDTIME(110000)
```

### ***Holding a job schedule entry***

The Hold Job Schedule Entry (HLDJOBSCDE) command allows you to hold an entry, all entries, or a set of entries in the job schedule. If an entry is held, no job is submitted at the scheduled time. To hold a job schedule entry, use the character-based interface.

To hold entries, you must have job control (\*JOBCTL) special authority; otherwise you can hold only those entries that you added. If you hold a job schedule entry:

- The entry is held until it is released using the Release Job Schedule Entry (RLSJOBSCDE) or Work with Job Schedule Entries (WRKJOBSCDE) command.
- The job is not submitted when it is released, even if a date and time at which it was scheduled to be submitted passed while the entry was held. Rather the job is submitted on any future dates for which it is scheduled to be submitted.

**Command:** Hold Job Schedule Entry (HDLJOBSCDE)

**Example:** The following example places the job schedule entry CLEANUP on hold.

```
HLDJOBSCDE JOB(CLEANUP)
```

### ***Printing a list of job scheduled entries***

To print a list of job schedule entries, use the character-based interface.

**Command:** Work with Job Schedule Entries (WRKJOBSCDE)

**Example:** The following example prints a list of job schedule entries.

```
WRKJOBSCDE OUTPUT(*PRINT)
```

**Example:** The following prints detailed information about each job schedule entry.

```
WRKJOBSCDE OUTPUT(*PRINT) PRTFMT(*FULL)
```

### ***Releasing a job schedule entry***

The Release Job Schedule Entry (RLSJOBSCDE) command allows you to release an entry, all entries, or a set of entries in the job schedule. If you release a job schedule entry, the job is not submitted immediately, even if the date and time at which it was scheduled to be submitted passed while the entry was held. If the scheduled time passed while the entry was held a warning message is sent to indicate that a job or jobs were missed. Then the job is submitted on any future dates for which it is scheduled to be submitted. To release job schedule entries, use the character-based interface.

To release entries, you must have job control (\*JOBCTL) special authority; otherwise you can release only those entries that you added.

**Command:** Release Job Schedule Entry (RLSJOBSCDE)

**Example:** This example release all job schedule entries that have a hold status.

```
RLSJOBSCDE JOB(*ALL) ENRYNBR(*ALL)
```

### ***Removing a job schedule entry***

The Remove Job Schedule Entry (RMVJOBSCDE) command allows you to remove an entry, entries, or generic entries in the job schedule. Each job schedule entry corresponds to one batch job, and contains the information needed to automatically run the job once or at regularly scheduled intervals. A message is sent to you and the message queue specified in the job schedule entry when an entry is successfully removed. To remove a job schedule entry, use the character-based interface.

To remove entries, you must be running under a user profile which has job control (\*JOBCTL) special authority; otherwise you can remove only those entries that you added.

**Command:** Remove Job Schedule Entry (RMVJOBSCDE)

**Example:** The following example removes the job PAYROLL from the job schedule.

When the system job removes a single-submission entry or when an entry is removed by the Remove Job Schedule Entry (RMVJOBSCDE) command, system message CPC1239 is sent to the message queue specified in the entry. If a single-submission entry was held when its schedule time was reached and the entry specified \*NO for its save attribute, the entry is removed when it is released with the Release Job Schedule Entry command. In this case, message CPC1245 is sent to the message queue specified in the entry.

## Managing subsystems

Because jobs run in subsystems, you might need to monitor subsystem activity for potential problems that can affect a job's ability to run.

The subsystem is the work place for jobs on your system. All user work is done by jobs running in the subsystem and it is important to monitor this area for slow work performance. In From IBM Navigator for i, 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.

### Common subsystem tasks

This information discuss the most common tasks that you can perform on a subsystem.

#### Related concepts

##### Subsystems

The subsystem is where work is processed on the system. A subsystem is a single, predefined operating environment through which the system coordinates the work flow and resource use. The system can contain several subsystems, all operating independently of each other. Subsystems manage resources.

#### Related information

Experience Report: Subsystem Configuration

### *Viewing subsystem attributes*

Subsystems have attributes. These attributes give information about the current status of the subsystem, or about values identified in the subsystem description.

When you use IBM Navigator for i, the following attributes 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

#### *IBM Navigator for i*

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

1. Expand **Work Management > Active Subsystems**.
2. Right-click the subsystem you want to view, then click **Properties**.

#### *Character-based interface*

To use the character-based interface, type in the following command:

**Command:** Display Subsystem Description (DSPSBSD)

**Example:** This command displays the subsystem description menu for the subsystem QBATCH.

## Stopping a subsystem

You can use IBM Navigator for i or the character-based interface 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, such as data that has been partially updated.

There are two types of stops.

### Controlled (Recommended)

Ends the subsystem in a controlled manner. The jobs are also ended in a controlled manner. This allows the programs that are running to perform cleanup (end of job processing). When a job being ended has a signal handling procedure for the asynchronous signal SIGTERM, the SIGTERM signal is generated for that job. The application has the amount of time specified for the DELAY parameter to complete cleanup before the job is ended.

### Immediate

Ends the subsystem immediately. The jobs are also ended immediately. When a job being ended has a signal handling procedure for the asynchronous signal SIGTERM, the SIGTERM signal is generated for that job and the QENDJOB LMT system value specifies a time limit. Other than handling the SIGTERM signal, the programs that are running are not allowed to perform any cleanup.

### Related concepts

[Ending a job: controlled](#)

Ending a job in a controlled manner allows programs that are running in the job to perform their end-of-job cleanup. A delay time can be specified to allow the job to end in a controlled manner. If the delay time ends before the job ends, the job is ended immediately.

### Related tasks

[How to display job logs](#)

You can see a job log from any place within work management that you access jobs, such as through the Subsystem area or the Memory Pool area. You can use IBM Navigator for i or the character-based interface to display job logs.

### Related information

[Jobs system values: Maximum time for immediate end](#)

*IBM Navigator for i*

To use IBM Navigator for i, use the following instructions:

1. Expand **Work Management > Active Subsystems**.
2. Right-click the subsystem or subsystems you want to stop, and then click **Stop**.
3. Specify the options to be used when the subsystem is stopped.
4. Click **Stop**.

*Character-based interface*

To use the character-based interface, type in the following command:

**Command:** End Subsystem (ENDSBS)

**Example:** This command ends all active jobs in the QBATCH subsystem and ends the subsystem. The active jobs are allowed 60 seconds to perform application-provided end-of-job processing.

```
ENDSBS SBS(QBATCH) OPTION(*CNTRLD) DELAY(60)
```

Use the End Subsystem Option (ENDSBSOPT) parameter to improve the performance for ending a subsystem. If you specify ENDSBSOPT(\*NOJOBLOG), the subsystem ends, but a job log is not produced for every job that was in the subsystem.

If a problem occurs in a job, but you specified \*NOJOBLOG, problem diagnosis may be difficult or impossible because the problem is not recorded in the job log. If you used the LOGOUTPUT(\*PND) job attribute then the job log is placed in a pending state, but is not written. However, the job log is still available if it is needed. See the related topics on job logs for more information about job log pending.

If you specify ENDSBSOPT(\*CHGPTY \*CHGTSL), the run priority and time slice change for all jobs that end in this subsystem. The jobs are competed less aggressively for processor cycles and are ended with less of an impact on jobs that are still running in other subsystems.

You can specify all three options (\*NOJOBLOG, \*CHGPTY, and \*CHGTSL) on the ENDSBSOPT parameter, for example:

```
ENDSBSOPT(*NOJOBLOG *CHGPTY *CHGTSL)
```

**Note:** If you specify \*ALL for the subsystem name and have any jobs running under QSYSWRK, you should use \*CNTRLD to prevent a subsystem from ending abnormally.

### **Starting a subsystem**

The Start Subsystem (STRSBS) command starts a subsystem using the subsystem description specified in the command. When the subsystem is started, the system allocates the necessary and available resources (storage, workstations, and job queues) that are specified in the subsystem description. You can start a subsystem by using IBM Navigator for i interface or the character-based interface.

#### **Related concepts**

How a subsystem starts

When a subsystem starts, the system allocates several items and starts autostart and prestart jobs before the subsystem is ready for work.

*IBM Navigator for i*

To start a subsystem using IBM Navigator for i, use the following instructions:

1. Expand **Work Management > Active Subsystems**.
2. Click **Actions > Start Subsystem**.
3. Indicate the **Name** and **Library** of the subsystem to start and click **OK**.

*Character-based interface*

**Command:** Start Subsystem (STRSBS)

**Example:** This command starts the user subsystem that is associated with the TELLER subsystem description in the QGPL library. The subsystem name is TELLER.

```
STRSBS SBSD(QGPL/TELLER)
```

### **Creating a subsystem description**

You can create a subsystem description in two ways. You can copy an existing subsystem description and change it, or you can create an entirely new description.

The following are two approaches that you can use:

1. To copy an existing subsystem description, using the character-based interface, use the following instructions:



- a) Create a Duplicate Object (CRTDUPOBJ) of an existing subsystem description. (You can also use the Work with Objects (WRKOBJ) or Work with Objects using Programming Development Manager (WRKOBJPDM) commands.)
  - b) Change the copy of the subsystem description so that it functions in the manner that you require. For example, you need to remove the job queue entry because it identifies the job queue that the original subsystem uses. Then you need to create a new job queue entry that specifies the parameters that the new subsystem uses.
 

Remember to review the autostart job entries, the workstation entries, the prestart job entries, and the communication entries, and verify that there are no conflicts between the two subsystems. For example, verify that the workstation entries do not cause both subsystems to allocate the same display devices.
2. To create an entirely new subsystem description, use the character-based interface and use the following instructions:
- a) Create a Subsystem Description (CRTSBSD).
  - b) Create a Job Description (CRTJOB).
  - c) Create a Class (CRTCLS) for Add Prestart Job Entry (ADDPJE) and Add Routing Entry (ADDRTGE).
  - d) Add Work Entries to the subsystem description.
    - Add Workstation Entry (ADDWSE)
    - Add Job Queue Entry (ADDJOBQE)
    - Add Communications Entry (ADDCMNE)
    - Add Autostart Job Entry (ADDAJE)
    - Add Prestart Job Entry (ADDPJE)
  - e) Add Routing Entries (ADDRTGE) to the subsystem description.

### Related concepts

#### Subsystems

The subsystem is where work is processed on the system. A subsystem is a single, predefined operating environment through which the system coordinates the work flow and resource use. The system can contain several subsystems, all operating independently of each other. Subsystems manage resources.

#### Subsystem description

A subsystem description is a system object that contains information defining the characteristics of an operating environment controlled by the system. The system-recognized identifier for the object type is \*SBSD. A subsystem description defines how, where, and how much work enters a subsystem, and which resources the subsystem uses to perform the work. An active subsystem takes on the simple name of the subsystem description.

### Related information

Experience Report: Subsystem Configuration

### ***Adding autostart job entries***

You use the character-based interface to add an autostart job entry. An autostart job starts automatically when the associated subsystem 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.

**Command:** Add Autostart Job Entry (ADDAJE)

**Example:** This example adds an autostart job entry to subsystem ABC's description.

```
ADDAJE SBSD(USERLIB/ABC) JOB(START)
      JOBD(USERLIB/STARTJD)
```

**Note:** For the changes to take effect, the active subsystem must be ended and then restarted.

## Related concepts

### [Autostart job entries](#)

Autostart job entries identify the autostart jobs to start as soon as the subsystem starts. When a subsystem starts, the system allocates several items and starts autostart and prestart jobs before it is ready for work.

## Adding communications entries

Each communication entry describes one or more communication device, device types, or remote location for which the subsystem starts jobs when program start requests are received. The subsystem can allocate a communication device if the device is not currently allocated to another subsystem or job. A communications device that is currently allocated may eventually be de-allocated, making it available to other subsystems. To add a communications entry to the subsystem description, use the character-based interface.

**Command:** Add Communications Entry (ADDCMNE)

**Example:** This example adds a communications entry for the APPC device named COMDEV and mode \*ANY to the subsystem description SBS1, which resides in library ALIB. The DFTUSR parameter defaults to \*NONE, which means that no jobs may enter the system through this entry unless valid security information is supplied on the program start request.

```
ADDCMNE  SBS1(ALIB/SBS1)  DEV(COMDEV)
```

**Note:** You must specify either the DEV parameter or the RMTLOCNAME parameter, but not both.

## Related concepts

### [Communications entries](#)

The communications work entry identifies to the subsystem the sources for the communications job it processes. The job processing begins when the subsystem receives a communications program start request from a remote system and an appropriate routing entry is found for the request.

## Adding job queue entries

A job queue entry identifies a job queue from which jobs are selected for running in the subsystem. Jobs started from a job queue are batch jobs. You add a job queue entry using the character-based interface.

You can specify the following items in a job queue entry.

- Job queue name (JOBQ)
- Maximum number of jobs that can be active at the same time from the job queue (MAXACT)
- Order in which the subsystem selects job queues from which jobs can be started (SEQNBR)
- Maximum number of jobs that can be active at the same time for a specified job queue priority (MAXPTYn)

**Command:** Add Job Queue Entry (ADDJOBQE)

**Example:** This command adds a job queue entry for the NIGHT job queue (in the QGPL library) to the NIGHTSBS subsystem description contained in the QGPL library. The entry specifies that up to three batch jobs from the NIGHT job queue can be active at the same time in the subsystem. The default sequence number of 10 is assumed.

```
ADDJOBQE  SBS1(QGPL/NIGHTSBS)  JOBQ(QGPL/NIGHT)  MAXACT(3)
```

## Related concepts

### [Job queue entry](#)

A job queue entry identifies a job queue from which jobs are selected for running in the subsystem. There are five parameters in the job queue entry that control how the job queue should be handled.

### [Job queue entries](#)

Job queue entries in a subsystem description specify from which job queues a subsystem is to receive jobs. When the subsystem is started, the subsystem tries to allocate each job queue defined in the subsystem job queue entries.

### ***Adding prestart job entries***

Prestart job entries identify prestart jobs that may be started when the subsystem is started or when the Start Prestart Jobs (STRPJ) command is entered. You can add prestart job entries to the subsystem description by using the character-based interface.

**Command:** Add Prestart Job Entry (ADDPJE)

**Example:** The following example adds a prestart job entry to the subsystem description ABC.

```
ADDPJE SBSDBS(USERLIB/ABC) PGM(START)
      JOBD(USERLIB/STARTPJ)
```

### **Related concepts**

#### Prestart job entries

You define the prestart job by using a prestart job entry. A prestart job entry does not affect the device allocation or program start request assignment.

#### Prestart job investigation

This topic provides steps to help you answer the question, "How do I find the real user of a prestart job and determine the resources used by that prestart job?"

### **Related information**

[Experience Report: Tuning prestart job entries](#)

### ***Adding routing entries***

Each routing entry specifies the parameters used to start a routing step for a job. Routing entries identify the main storage subsystem pool to use, the controlling program to run (typically the system-supplied program QCMD), and additional run-time information (stored in the class object). To add a routing entry to a subsystem description, use the character-based interface.

**Command:** Add Routing Entry (ADDRTGE)

**Example:** This command adds routing entry 46 to the subsystem description PERT in the ORDLIB library. To use routing entry 46, the routing data must start with the character string WRKSTN2 starting in position 1. Any number of routing steps can be active through this entry at any one time. The program GRAPHIT in the library ORDLIB is to run in storage pool 2 by using class AZERO in library MYLIB.

```
ADDRTGE SBSDBS(ORDLIB/PERT) SEQNBR(46) CMPVAL(WRKSTN2)
      PGM(ORDLIB/GRAPHIT) CLS(MYLIB/AZERO) MAXACT(*NOMAX)
      POOLID(2)
```

### **Related concepts**

#### Routing entries

The routing entry identifies the main storage subsystem pool to use, the controlling program to run (typically the system-supplied program QCMD), and additional run-time information (stored in the class object). Routing entries are stored in the subsystem description.

### ***Adding workstation entries***

A workstation entry is used when a job is started when a user signs on or transfers an interactive job from another subsystem. You can specify the following items in a workstation entry. Parameter names are given in parentheses. Use the character-based interface to add workstation entries.

- Workstation name or type (WRKSTN or WRKSTNTYPE)
- Job description name (JOBD) or job description name in the user profile
- Maximum number of jobs that can be active at the same time through the entry (MAXACT)
- When the workstations are to be allocated, either when the subsystem is started or when an interactive job enters the subsystem through the Transfer Job (TFRJOB) command and the AT parameter.

To add a workstation entry to a subsystem description, use the character-based interface.

**Command:** Add Workstation Entry (ADDWSE)

**Example:** The following example adds the workstation entry DSP12 to the subsystem ABC.

```
ADDWSE SBSDB(USERLIB/ABC) WRKSTN(DSP12)
      JOBD(USERLIB/WSE)
```

### Related concepts

#### Workstation entries

An interactive job is a job that starts when a user signs on to a display station and ends when the user signs off. For the job to run, the subsystem searches for the job description, which may be specified in the workstation entry or the user profile.

### ***Creating a sign-on display file***

The sign-on display file is used to show sign-on displays at workstations that are allocated to the subsystem. The sign-on display file can be changed when the subsystem is active. However, the new sign-on display file is not used until the next time the subsystem is started. To create a sign-on display file, use the character-based interface.

A new sign-on display file can be created using the IBM-supplied sign-on display file as a starting point. The source for this display file is located in library QGPL in source physical file QDDSSRC. It is strongly recommended that you create a new source physical file and copy the IBM-supplied display file to the new source physical file before making any changes. In this way, the original IBM-supplied source is still available.

### **Considerations:**

- The order in which the fields in the sign-on display file are declared must not be changed. The position in which they are displayed on the display can be changed.
- Do not change the total size of the input or output buffers. Serious problems can occur if the order or size of the buffers are changed.
- Do not use the data descriptions specifications (DDS) help function in the sign-on display file.
- Always specify 256 on the MAXDEV parameter.
- The MENUBAR and PULLDOWN keywords cannot be specified in a sign-on display file description.
- The buffer length for the display file must be 318. If it is less than 318, the subsystem uses the default sign-on display, QDSIGNON in library QSYS.
- The copyright line cannot be deleted.
- Member QDSIGNON is the IBM-supplied sign-on display file that utilizes a 10 character password.
- Member QDSIGNON2 is the IBM-supplied sign-on display file that utilizes a 128 character password.

**Command:** Create Display File (CRTDSPF)

A hidden field in the display file named UBUFFER can be changed to manage smaller fields. UBUFFER is 128 bytes long and is stated as the last field in the display file. This field can be changed to function as an input/output buffer so that the data specified in this field is available to application programs when the interactive job is started. You can change the UBUFFER field to contain as many smaller fields as you need if the following requirements are met:

- The new fields must follow all other fields in the display file. The location of the fields on the display does not matter as long as the order in which they are put in the data description specifications (DDS) meets this requirement.
- The length must total 128. If the length of the fields is more than 128, some of the data is not passed.
- All fields must be input/output fields (type B in DDS source) or hidden fields (type H in DDS source).

### **Related information**

[Locales as part of a multi-lingual environment](#)

[DDS for display files](#)

## Specifying the new sign-on display

A subsystem uses the sign-on display file that is specified in the SGNDSPF parameter of the subsystem description to create the sign-on display at a user workstation. To change the sign-on display file from the default (QDSIGNON) to one that you have created, use the character based interface.

**Note:** Use a test version of a subsystem to verify that the display is valid before attempting to change the controlling subsystem.

**Command:** Change Subsystem Description (CHGSBSD)

Specify the new display file on the SGNDSPF parameter.

**Example:** The following changes the sign-on display file for subsystem QBATCH from the default to a new file called MYSIGNON.

```
CHGSBSD SBSD(QSYS/QBATCH) SGNDSPF(MYSIGNON)
```

### Related information

[Locales as part of a multi-lingual environment](#)

[DDS for display files](#)

## Changing a subsystem description

The Change Subsystem Description (CHGSBSD) command changes the operational attributes of the specified subsystem description. You can change the subsystem description while the subsystem is active. To change a subsystem description, use the character based interface.

**Note:** You cannot specify the \*RMV value on the POOLS parameter while the subsystem is active, because a job might become suspended.

**Command:** Change Subsystem Description (CHGSBSD)

**Example:** This command changes the definition of storage pool 2 that is used by subsystem PAYCTL to a storage size of 1500K and an activity level of 3. The sign-on display file is changed to display file COMPANYYA and is located in the QGPL library. If the subsystem is active when this command is issued, COMPANYYA is not used until the next time the subsystem is started.

```
CHGSBSD SBSD(QGPL/PAYCTL) POOLS((2 1500 3))
SGNDSPF(QGPL/COMPANYYA)
```

### Related concepts

[Subsystem description](#)

A subsystem description is a system object that contains information defining the characteristics of an operating environment controlled by the system. The system-recognized identifier for the object type is \*SBSD. A subsystem description defines how, where, and how much work enters a subsystem, and which resources the subsystem uses to perform the work. An active subsystem takes on the simple name of the subsystem description.

## Changing autostart job entries

You can specify a different job description for a previously defined autostart job entry. To change an autostart job entry, use the character-based interface

**Command:** Change Autostart Job Entry (CHGAJE)

**Example:** The following example changes the job description used by the autostart job entry named START, in subsystem ABC, in library USERLIB.

```
CHGAJE SBSD(USERLIB/ABC) JOB(START)
JOB(D(USERLIB/NEWJD))
```

**Note:** For the changes to take effect, the active subsystem must be ended and then restarted.

## Related concepts

### Autostart job entries

Autostart job entries identify the autostart jobs to start as soon as the subsystem starts. When a subsystem starts, the system allocates several items and starts autostart and prestart jobs before it is ready for work.

## Changing communication entries

You can change the attributes of an existing communications entry in an existing subsystem description by using the character-based interface.

- When the Job description (JOBBD) or Default user profile (DFTUSR) parameters are changed, the communications entry also changes; however, the values of these parameters are not changed for any jobs that are active at the time.
- If the value of the Maximum active jobs (MAXACT) parameter is reduced to a number less than the total number of jobs that are active through the communications entry, no new program start requests are processed. Active jobs continue to run; but no additional program start requests are processed until the number of active jobs is less than the value specified for the MAXACT parameter.

**Command:** Change Communications Entry (CHGCMNE)

**Example:** This example changes the communications entry (in the subsystem description QGPL/BAKER) for the device A12 and mode \*ANY. The maximum activity level is changed to \*NOMAX which means that the communications entry puts no restrictions on the number of program start requests that might be active at the same time. However, the MAXJOBS value in the subsystem description BAKER limits the total number of jobs that can be active in the subsystem. This includes those created by program start requests. There is also a limit that the user can specify on the number of active jobs that can be routed through any particular routing entry (MAXACT). The limit specified in the routing entry can control the number of jobs using a particular pool or the recursion level of a particular program. In all cases, none of these limits can be exceeded as a result of processing a program start request.

```
CHGCMNE  SBS(D(QGPL/BAKER)  DEV(A12)  MAXACT(*NOMAX)
```

## Related concepts

### Communications entries

The communications work entry identifies to the subsystem the sources for the communications job it processes. The job processing begins when the subsystem receives a communications program start request from a remote system and an appropriate routing entry is found for the request.

## Changing job queue entries

You can change an existing job queue entry in the specified subsystem description. This command can be issued while a subsystem is active or inactive. To change the job queue entry in a subsystem, use the character-based interface.

**Command:** Change Job Queue Entry (CHGJOBQE)

**Example:** This command changes the maximum number of jobs that can be active at the same time from the job queue QBATCH in library QGPL. The sequence number of the job queue entry does not change. Up to four jobs from the QBATCH job queue can be active at the same time. Up to one job can be active from priority level 1. There is no maximum for the number of jobs that can be active at the same time from priority level 2. Priority levels 3 through 9 do not change.

```
CHGJOBQE  SBS(D(QGPL/QBATCH)  JOBQ(QGPL/QBATCH)  MAXACT(4)
           MAXPTY1(1)  MAXPTY2(*NOMAX)
```

## Related concepts

### Job queue entry

A job queue entry identifies a job queue from which jobs are selected for running in the subsystem. There are five parameters in the job queue entry that control how the job queue should be handled.

### Job queue entries

Job queue entries in a subsystem description specify from which job queues a subsystem is to receive jobs. When the subsystem is started, the subsystem tries to allocate each job queue defined in the subsystem job queue entries.

### ***Changing prestart entries***

You can change a prestart job entry in the specified subsystem description. The subsystem may be active when the prestart job entry is changed. Changes made to the entry when the subsystem is active are reflected over time. Any new prestart jobs started after the command is issued use the new job-related values. This command identifies prestart jobs that are started when the subsystem is started or when the Start Prestart Jobs (STRPJ) command is issued.

To change the prestart job entry of a subsystem description, use the character-based interface.

**Command:** Change Prestart Job Entry (CHGPJE)

**Example:** This example changes the prestart job entry for the PGM1 program in the QGPL library in the PJSBS subsystem description contained in the QGPL library. The prestart jobs associated with this entry are not started the next time the PJSBS subsystem description in the QGPL library is started. The STRPJ command is needed to start the prestart jobs. When more jobs need to be started, one additional job is started.

```
CHGPJE  SBSDB(QGPL/PJSBS)  PGM(QGPL/PGM1)  STRJOBS(*NO)
        THRESHOLD(1)  ADLJOBS(1)
```

### **Related concepts**

[Prestart job entries](#)

You define the prestart job by using a prestart job entry. A prestart job entry does not affect the device allocation or program start request assignment.

[Prestart job investigation](#)

This topic provides steps to help you answer the question, "How do I find the real user of a prestart job and determine the resources used by that prestart job?"

### **Related information**

[Experience Report: Tuning prestart job entries](#)

### ***Changing routing entries***

You can change a routing entry in the specified subsystem description by using the character-based interface. The routing entry specifies the parameters used to start a routing step for a job. The associated subsystem can be active when the changes are made.

**Command:** Change Routing Entry (CHGRTGE)

**Example:** This command changes routing entry 1478 in the subsystem description ORDER found in library LIB5. The same program is used, but now it runs in storage pool 3 using class SOFAST in library LIB6.

```
CHGRTGE  SBSDB(LIB5/ORDER)  SEQNBR(1478)  CLS(LIB6/SOFAST)  POOLID(3)
```

### **Related concepts**

[Routing entries](#)

The routing entry identifies the main storage subsystem pool to use, the controlling program to run (typically the system-supplied program QCMD), and additional run-time information (stored in the class object). Routing entries are stored in the subsystem description.

### ***Changing workstation entries***

You can specify a different job description for a previously defined workstation entry by using the character-based interface.

- When the Job description (JOBID) parameter is specified, the workstation entry is changed; however, the value of this parameter is not changed for any jobs started through this entry that are active at the time.



- If the value of the Maximum active jobs (MAXACT) parameter is reduced to a number less than the total number of workstations that are active through the workstation entry, no additional workstations are allowed to sign on. Active workstations are signed off. Additional jobs can be created for an active workstation by the Transfer Secondary Job (TFRSECJOB) command or the Transfer to Group Job (TFRGRPJOB) command. Other workstations are not allowed to sign on until the number of active workstations is less than the value specified for the MAXACT parameter.

**Command:** Change Workstation Entry (CHGWSE)

**Example:** This command changes the workstation entry for workstation A12 in subsystem BAKER found in the general purpose library. A job is created for workstation A12 when the user's password is entered on the sign-on display and the Enter key is pressed.

```
CHGWSE  SBSDB(QGPL/BAKER)  WRKSTN(A12)  AT(*SIGNON)
```

## Related concepts

### Workstation entries

An interactive job is a job that starts when a user signs on to a display station and ends when the user signs off. For the job to run, the subsystem searches for the job description, which may be specified in the workstation entry or the user profile.

## ***Changing the sign-on display***

Your system is shipped with the default sign-on display file of QDSIGNON, which is located in the QSYS library. In situations where you have a multilingual environment, you might want to change what is displayed on the sign-on screen. Or, perhaps you want to add your company information to the sign-on screen. In such situations you must first create a new display file. To do so, use the character-based interface.

The SGNDSPF attribute in the subsystem description points to the sign-on display file that the user sees when signing on to the subsystem.

The steps used to change the sign-on display are summarized as follows:

1. Create a new sign-on display file.
2. Change the subsystem description to use the changed display file instead of the system default.
3. Test the change.

## Related tasks

### Creating a sign-on display file

The sign-on display file is used to show sign-on displays at workstations that are allocated to the subsystem. The sign-on display file can be changed when the subsystem is active. However, the new sign-on display file is not used until the next time the subsystem is started. To create a sign-on display file, use the character-based interface.

### Specifying the new sign-on display

A subsystem uses the sign-on display file that is specified in the SGNDSPF parameter of the subsystem description to create the sign-on display at a user workstation. To change the sign-on display file from the default (QDSIGNON) to one that you have created, use the character based interface.

## Related information

### Locales as part of a multi-lingual environment

### DDS for display files

## **Deleting a subsystem description**

The Delete Subsystem Description (DLTSBSD) command deletes the specified subsystem descriptions (including any work entries or routing entries added to them) from the system. Job queues assigned to this subsystem by the Add Job Queue Entry (ADDJOBQE) command are not deleted. In fact, when you delete a subsystem description (SBSD), none of the objects that are referenced by the SBSDB are deleted.

The associated subsystem must be inactive before it can be deleted. Use the character-based interface to delete a subsystem.



**Command:** Delete Subsystem Description (DLTSBSD)

This command deletes the inactive subsystem description called BAKER from library LIB1.

```
DLTSBSD  SBSB(LIB1/BAKER)
```

### ***Removing autostart job entries***

You can remove an autostart job entry from a subsystem description by using the character-based interface.

**Command:** Remove Autostart Job Entry (RMVAJE)

**Example:** The following example removes the autostart entry for the job START from the subsystem description ABC.

```
RMVAJE  SBSB(USERLIB/ABC)  JOB(START)
```

**Note:** For the changes to take effect the active subsystem must be ended and then restarted.

### **Related concepts**

[Autostart job entries](#)

Autostart job entries identify the autostart jobs to start as soon as the subsystem starts. When a subsystem starts, the system allocates several items and starts autostart and prestart jobs before it is ready for work.

### ***Removing communication entries***

You can remove communication entries from the subsystem description by using the character-based interface. All jobs that are active through the communications entry being removed must be ended before this command can be run.

**Command:** Remove Communications Entry (RMVCMNE)

**Example:** This command removes the communications device entry for the device COMDEV from the subsystem description SBS1 in library LIB2.

```
RMVCMNE  SBSB(LIB2/SBS1)  DEV(COMDEV)
```

### **Related concepts**

[Communications entries](#)

The communications work entry identifies to the subsystem the sources for the communications job processes. The job processing begins when the subsystem receives a communications program start request from a remote system and an appropriate routing entry is found for the request.

### ***Removing job queue entries***

You can remove job queue entries from a subsystem description by using the character-based interface. Jobs on the job queue remain on the queue when the job queue entry is removed from the subsystem description. A job queue entry cannot be removed if any currently active jobs were started from the job queue.

**Command:** Remove Job Queue Entry (RMVJOBQE)

**Example:** This command removes the job queue entry that refers to the BATCH2 job queue in MYLIB from the NIGHTRUN subsystem description stored in library MYLIB.

```
RMVJOBQE  SBSB(MYLIB/NIGHTRUN)  JOBQ(MYLIB/BATCH2)
```

### **Related concepts**

[Job queue entry](#)

A job queue entry identifies a job queue from which jobs are selected for running in the subsystem. There are five parameters in the job queue entry that control how the job queue should be handled.

[Job queue entries](#)

Job queue entries in a subsystem description specify from which job queues a subsystem is to receive jobs. When the subsystem is started, the subsystem tries to allocate each job queue defined in the subsystem job queue entries.

### **Related tasks**

[Assigning the job queue to the subsystem](#)

To assign a job queue entry to a subsystem description, use the character-based interface.

### ***Removing prestart job entries***

You can remove prestart job entries from the subsystem description by using the character-based interface. A prestart job entry cannot be removed if any currently active jobs were started using the entry.

When removing an entry where \*LIBL is specified for the library name, the library list is searched for a program with the specified name. If a program is found in the library list but an entry exists with a different library name (which is found later in the library list), no entry is removed. If a program is not found in the library list but an entry exists, no entry is removed.

**Command:** Remove Prestart Job Entry (RMVPJE)

**Example:** This command removes the prestart job entry for the PGM1 program (in the QGPL library) from the PJE subsystem description contained in the QGPL library.

```
RMVPJE  SBS(D(QGPL/PJE)  PGM(QGPL/PGM1)
```

### **Related concepts**

[Prestart job entries](#)

You define the prestart job by using a prestart job entry. A prestart job entry does not affect the device allocation or program start request assignment.

[Prestart job investigation](#)

This topic provides steps to help you answer the question, "How do I find the real user of a prestart job and determine the resources used by that prestart job?"

### **Related information**

[Experience Report: Tuning prestart job entries](#)

### ***Removing routing entries***

You can remove a routing entry from the specified subsystem description by using the character-based interface. The subsystem can be active at the time the command is run. However, the routing entry cannot be removed if there are any currently active jobs that were started using the entry.

**Command:** Remove Routing Entry (RMVRTGE)

**Example:** This command removes the routing entry 9912 from subsystem description PERT in library OR.

```
RMVRTGE  SBS(D(OR/PERT)  SEQNBR(9912)
```

### **Related concepts**

[Routing entries](#)

The routing entry identifies the main storage subsystem pool to use, the controlling program to run (typically the system-supplied program QCMD), and additional run-time information (stored in the class object). Routing entries are stored in the subsystem description.

### ***Removing workstation entries***

You can remove a workstation entry from a subsystem description by using the character-based interface. The subsystem can be active at the time the command is run. However, all jobs that are active through the workstation entry must be ended before it can be removed.

**Command:** Remove Work Station Entry (RMVWSE)

**Example:** This example removes the workstation entry for workstation B53 from the subsystem description named CHARLES in library LIB2.

**Related concepts**Workstation entries

An interactive job is a job that starts when a user signs on to a display station and ends when the user signs off. For the job to run, the subsystem searches for the job description, which may be specified in the workstation entry or the user profile.

**Configuring an interactive subsystem**

The information in this section explains how to set up a new interactive subsystem.

These steps are described as if the commands are entered manually. However, you can easily re-create your configurations for recovery purposes by using a CL program to create your subsystems.

When you set up a new interactive subsystem you should consider how many devices will be allocated to that subsystem. Since the subsystem performs device management functions, such as presenting the sign-on display and handling device error recovery, you might want to limit the number of devices allocated to a single subsystem. See the [Communications limits](#) topic for more information.

**Note:** This topic provides a synopsis of what is involved in configuring interactive subsystems. The experience reports about subsystems contain detailed explanations of each step and additional options available for each step.

**Creating a library**

This example shows how to create a library to store your subsystem configuration objects.

The example uses SBSLIB as the library.

```
CRTLIB SBSLIB TEXT('LIBRARY TO HOLD SUBSYSTEM CONFIGURATION OBJECTS')
```

**Creating a class**

A class defines certain performance characteristics for your interactive subsystem. Follow this instruction to create a class.

To create a class that is identical to the QINTER class, enter the following command:

```
CRTCLS SBSLIB/INTER1 RUNPTY(20) TIMESLICE(2000) PURGE(*YES) DFTWAIT(30)
TEXT('Custom Interactive Subsystem Class')
```

You can use the QINTER class in QGPL for your custom interactive subsystems, or you can create a single class to use for all of your interactive subsystems, or you can create one for each interactive subsystem.

Your choice should depend upon whether you want to customize some of the performance settings for a particular subsystem. IBM-supplied subsystems are shipped with a class created for each subsystem, with the name of the class being the same as the name of the subsystem.

If you do NOT create a class for each subsystem with the same name as the subsystem, you need to specify the class name on the Add Routing Entry (ADDRTGE) command. This is because the default for the CLS parameter is \*SBSD, meaning the class name has the same name as the subsystem description.

**Creating the subsystem description**

For each subsystem that you need to define, repeat this step to create the subsystem description.

The following creates a subsystem description with attributes identical to those of QINTER.

```
CRTSBSD SBSDB(SBSLIB/INTER1) POOLS((1 *BASE) (2 *INTERACT)) SGNDSPF(*QDSIGNON)
```

## Creating a job queue

You can create a job queue for the subsystem using the same name as the subsystem name and add a job queue entry to the subsystem description.

This step is required if you need to use the Transfer Job (TFRJOB) command to transfer jobs into your custom subsystems.

```
CRTJOBQ JOBQ(SBSLIB/INTER1)
ADDJOBQE SBSDB(SBSLIB/INTER1) JOBQ(SBSLIB/INTER1) MAXACT(*NOMAX)
```

## Adding a routing entry

The routing entries that are supplied with the system for QINTER have some additional functions. If you need those functions, add those routing entries to your customized subsystem descriptions.

Follow this step to add a routing entry:

```
ADDRTGE SBSDB(SBSLIB/INTER1) SEQNBR(9999) CMPVAL(*ANY) PGM(QSYS/QCMD) POOLID(2)
```

## Adding workstation entries

Adding workstation entries to the subsystem description is a key step for assigning which devices are allocated to which subsystem.

You need to determine which subsystems should allocate which devices (AT(\*SIGNON)). In addition, determine if you need to allow the use of TFRJOB from one subsystem to another (AT(\*ENTER)).

```
ADDWSE SBSDB(SBSLIB/PGRM) WRKSTN(PGMR*) AT(*SIGNON)
ADDWSE SBSDB(SBSLIB/ORDERENT) WRKSTN(ORDERENT*) AT(*SIGNON)
ADDWSE SBSDB(QGPL/QINTER) WRKSTN(QPADEV*) AT(*SIGNON)
```

In this example, the subsystem and device naming convention is based upon the type of work the user does. Programmers all have devices that are named with PGMR and run in the PGRM subsystem. Order entry personnel all have devices that are named with ORDERENT and run in the ORDERENT subsystem. All other users use the system default naming convention of QPADEVxxxx and run in the IBM-supplied subsystem QINTER.

## Customizing QINTER

When you begin using your own set of subsystems, you might not need to use QINTER. However, if you have a reason to continue to use QINTER, you need to ensure that QINTER is set up NOT to allocate the workstations that you want to run under your other subsystems. There are two possible ways to do this.

Remove the \*ALL workstation entry from QINTER:

1. Remove the \*ALL workstation entry from QINTER, and then add specific workstation entries that indicate which devices you want QINTER to allocate.

Removing the workstation type entry of \*ALL is to prevent QINTER from attempting to allocate all workstations.

2. Add a workstation entry for devices named DSP\* to allow all twinax-attached display devices to continue to be allocated to QINTER.

In this example, the twinax-attached display devices will continue to run in QINTER; QINTER will not attempt to allocate any other devices.

```
RMVWSE SBSDB(QGPL/QINTER) WRKSTNTYPE(*ALL)
ADDWSE SBSDB(QGPL/QINTER) WRKSTN(DSP*)
```

## Second method

Add a workstation entry to tell QINTER not to allocate the devices that are assigned to other subsystems. However, allow QINTER to continue to allocate any other device that is not allocated to a subsystem. This keeps the workstation type entry of \*ALL in the QINTER subsystem and adds workstation name entries with the AT parameter for those devices that are allocated to different subsystems.

```
ADDWSE SBS(D(QGPL/QINTER) WRKSTN(PGMR*) AT(*ENTER)
ADDWSE SBS(D(QGPL/QINTER) WRKSTN(ORDERENT*) AT(*ENTER)
```

**Note:** You cannot use this method if the number of device descriptions on your system exceeds the maximum number that a single subsystem can handle.

### **Configuring the console**

A final, but VERY important consideration regarding QINTER is the workstation type entry of \*CONS for the console. Make sure that you do not accidentally prevent someone from signing on at the console. You prevent this from happening by not adding any workstation entries for the console to your custom interactive subsystems.

The system is shipped with the controlling subsystem having a workstation entry of AT(\*SIGNON) for the console (\*CONS workstation type entry). QINTER has the AT(\*ENTER) workstation type entry for the console.

It is good practice to always run the console in the controlling subsystem and to not transfer the console job into some other interactive subsystem. This prevents the user at the console from ending their own job unintentionally.

For example, if the user at the console transfers their job to INTER1 and forgets about it, and sometime later proceeds to prepare for backup processing by doing an End System (ENDSYS) command, the console job is also ended. This is most likely not what the operator intended.

### **Assigning users to a specific subsystem**

You can use several techniques to assign device names and then associate those device names with users. After this is accomplished, you can use the workstation entries to get the user to the correct subsystem.

The system has a default naming convention that is used for display sessions. At times this proves insufficient for routing workstation entries across multiple subsystems by user profile.

You can make changes on your system to enhance the system's default behavior by assigning and managing your own device naming conventions. There are several ways of doing this. Each approach has its own set of advantages and disadvantages.

#### **Related concepts**

[How workstation devices are allocated](#)

Subsystems attempt to allocate all workstation devices in its subsystem description for AT(\*SIGNON) workstation entries.

#### **Related information**

[Experience Report: Subsystem Configuration](#)

[Using Telnet exit point programs](#)

#### *Telnet device initialization and terminal exit points*

Telnet Device Initialization and Terminal Exit Points. These exit points provide the ability to assign device names based upon the client signing on to the system.

The exit point provides you with the client IP address and the user profile name (along with additional information). You can then perform your own mapping of the client to the device description that should be used for the client.

The device initialization exit point also provides a way to bypass the sign-on panel.

The advantage to using these exit points to manage your device naming convention is that you have central control on your system for all of your clients.

The disadvantage is that it requires programming skills.

### *Device selection exit point*

This exit point allows you to specify the naming convention used for automatically created virtual devices and virtual controllers and to specify the automatic creating limit used for the special requests.

With this exit point you can specify different naming conventions for automatically created devices used by Telnet, 5250 Display Station Pass-through, and the virtual terminal APIs.

In addition, you can manage the Pass-through devices and Telnet (QAUTOVRT) system value in a more precise manner. For example, you can allow one value for automatically created devices for Telnet and allow a different value for 5250 Display Station Pass-through devices.

This exit point gives you the ability to control the default naming conventions used for devices (such as QPADEV\*) but it alone does not allow you to specify a particular device for a particular user. This exit point is most useful if you are using a mix of ways to connect to the system (Telnet, 5250 Display Station Pass-through, WebFacing, and so on) because it allows you to use different device naming conventions and precise QAUTOVRT management for different access methods.

### *PC5250 (System i Access) workstation ID support*

You can configure IBM i Access to connect with a specific workstation name. If you click the help button from this window, the various options for specifying the workstation ID, like generating a new name if the one specified is already in use, are displayed.

A disadvantage to this approach is that it requires you to manage the PC5250 configuration settings on each and every client that connects to your server.

### *OS/400 Telnet Client*

Using the OS/400® Telnet Client command (STRTCPTELN or TELNET), you can specify the device name that is used to sign on to the server system.

A disadvantage to the default approach is that it requires you to ensure that all usage of the STRTCPTELN (TELNET) commands specify the remote virtual display value appropriately. To alleviate this concern, you can create a custom version of the STRTCPTELN command to ensure the remote virtual terminal display value and start the IBM supplied command.

### *Manually creating virtual controllers and devices*

You can manually create your virtual controllers and devices.

For more information about virtual device creation for Telnet, see the Configure the Telnet Server topic in the IBM i Information Center.

This gives you control over what the names of your controllers and devices are, but it does not provide you with the ability to map a specific device to a specific user.

## **Configuring a server subsystem**

The information in this section explains how to set up a subsystem for server jobs.

The default subsystem configuration shipped with IBM i is a basic subsystem configuration that works well for small systems. However, as the number of users and amount of work increases on the system, you may want to split the work into multiple subsystems to better manage the work on the system.

Server jobs are jobs that run continuously in the background waiting for work. Work can come from network functions, operating system functions, on behalf of a user, another system within the network, or from general system services, such as the clustering server jobs. Server jobs typically run in one of the basic subsystems that are shipped with the system - QSYSWRK, QSERVER, or QUSRWRK. The daemon jobs (the server jobs that route work requests to waiting prestart jobs) typically run in QSYSWRK and QSERVER, while the prestart server jobs that perform work on behalf of users typically run in QUSRWRK, although some of the prestart jobs also run in QSERVER.

The following sections show how to create a user-defined server subsystem and customize the subsystems in which the server jobs run. Server jobs can be routed to a subsystem by IP address, or in some cases, by user name.

## Creating a user-defined server subsystem

A server can use the default subsystem, one of the subsystems that are shipped with the system, a user-defined server subsystem or a combination of one or more of these different types of subsystems. This section explains how to create your own server subsystem.

To create a user-defined server subsystem, follow these steps:

1. Create a library to store your subsystem configuration objects in.

```
CRTLIB SBSLIB TEXT('Library to hold subsystem configuration objects')
```

2. Create a class. The class defines certain performance characteristics for your subsystem including run priority, time slice, and default wait times.

```
CRTCLS SBSLIB/MYSBS RUNPTY(20) TIMESLICE(2000) DFTWAIT(30)  
TEXT('Custom Subsystem Class')
```

3. Create the subsystem description. You can either use the Create Subsystem Description (CRTSBSD) command to create a new subsystem description or the Create Duplicate Object (CRTDUPOBJ) command to make a copy of an existing subsystem description. For example, to create a new subsystem description:

```
CRTSBSD SBSDB(SBSLIB/MYSBS) POOLS((1 *BASE))  
TEXT('Custom Server Subsystem')
```

To create a duplicate of the QUSRWRK subsystem description:

```
CRTDUPOBJ OBJ(QUSRWRK) FROMLIB(QSYS) OBJTYPE(*SBSD) TOLIB(SBSLIB)
```

**Note:** Create duplicate object will also copy job queue entries, routing entries, and prestart job entries from the subsystem description being copied from. Depending on the requirements for your subsystem, these values may have to be changed.

4. Create a job queue for the subsystem, using the same name as the subsystem name and add a job queue entry to the subsystem description. In general, a job queue for the subsystem will be required. Having a job queue allows server jobs to be submitted to the subsystem when needed. Server jobs on the job queue will then be processed. Also, if you will be using the Transfer Job (TFRJOB) command to transfer jobs into your custom subsystem, you will want a job queue.

```
CRTJOBQ JOBQ(SBSLIB/MYSBS)  
ADDJOBQE SBSDB(SBSLIB/MYSBS) JOBQ(SBSLIB/MYSBS) MAXACT(*NOMAX)
```

**Note:** A job queue can only be allocated by one active subsystem. If you used CRTDUPOBJ to create your subsystem description in step 3, remove the JOBQ entries that were copied, using the Remove Job Queue Entry (RMVJOBQE) command, to prevent your subsystem from trying to allocate a JOBQ that is needed by the subsystem the entry was copied from.

5. Add a routing entry to the subsystem. The following is an example of adding a routing entry that is designed to be a generic entry to catch all requests that do not match a specific routing entry. The routing entries added to any user-defined subsystem can vary from this example and will depend on the specific requirements for the subsystem. A good basis for setting up routing entries for a user-defined subsystem is to make them identical to the routing entries used by QUSRWRK.

```
ADDRTGE SBSDB(SBSLIB/MYSBS) SEQNBR(9999) CMPVAL(*ANY)  
PGM(QSYS/QCMD) CLS(SBSLIB/MYSBS)
```

If you look at the routing entries shipped on the system for QUSRWRK, you will see there are several additional routing entries. If you need those functions, add those routing entries to your customized subsystem descriptions as well. It is recommended to copy the routing entries from QUSRWRK into the new user-defined subsystem.

**Note:** The maximum activity level (MAXACT) refers to the maximum number of routing steps that can be active through this routing entry. It is highly recommend to use the default value of \*NOMAX. The ADDRTGE command specifies which pool identifier to use and the default on the command is 1. If you

had set up your subsystem description with dedicated pools, be sure to specify the appropriate pool identifier on the routing entry. If the name of your class is different from the name of your subsystem description, you will need to specify the class on the ADDDRTGE command.

6. Add prestart job entries to the subsystem. A prestart job is a job that is started and waits for work to be dispatched to it. It is particularly useful for subsystems to have a number of prestart jobs available to handle requests that are submitted to the subsystem. The following is an example of adding a prestart job entry. The prestart job entries added to any user-defined subsystem can vary from this example and will depend on the specific requirements for the subsystem. It is recommended that the prestart job entries for a user-defined subsystem be identical to the prestart job entries used by QUSRWRK.

```
ADDPJE SBSDB(SBSLIB/MYSBS) PGM(QSYS/QZSOSIGN) INLJOBS(50) THRESHOLD(4)
      JOB(QZSOSIGN) JOBD(QSYS/QZBSJOB) CLS(QGPL/QCASERVR) STRJOBS(*YES)
```

**Note:** By initializing the job before work arrives, the overhead and time of doing initialization is avoided, allowing for a much higher throughput of work. It is very important that the prestart job entries be configured correctly to be able to handle the expected amount of work and the expected amount of requests sent to them. If not done correctly, jobs may end up taking an alternative action or a degradation in performance will be seen as additional prestart jobs are created if the current pool of prestart jobs is exhausted.

### ***Routing server jobs based on client IP address***

Some of the servers provided by IBM i can be configured to run in a subsystem other than the default subsystem. A server can use subsystems that are shipped with the system or user-defined server subsystems. A server can also use multiple subsystems. The following steps show how to configure server subsystems using IBM Navigator for i.

To specify which subsystem server jobs will use:

1. Select **Network > Servers**.
2. Select the type of server that you want to configure. DDM and IBM i NetServer are TCP/IP servers. Central, Database, Data Queue, Files, Net Print, Remote Command, and Sign On are IBM i Access servers. The remaining steps use a subsystem from IBM i Access. The subsystem configuration steps are identical for each server.
3. In the right pane, right-click on the server that you want to configure and select **Properties**, then switch to the **Subsystems** tab.
4. From the **Subsystems** tab you can specify which subsystem you want this server's jobs to run in. There are different ways to configure the subsystems. Select the appropriate configuration method and complete the following instructions for your selection:

- **User server defaults**

Select this option if you want the server to use the default subsystem for all jobs using this subsystem.

- **All clients**

Select this option if all clients that use this server are to use the same subsystem and alternate action.

**All clients** provides an easy way to subdivide the work done by the various servers into a subsystem by type of server. For example, you can have all clients for the database server use the DATABASE subsystem, all clients for the remote command server use the RMTCMD subsystem, etc.

- The **Subsystem** list specifies the subsystem you want this server's jobs to run in. You can type in the ten character name of the subsystem you want to use, if it is not listed. If None is selected, the server jobs will perform the **Alternate action**.
- The **Alternate action** list specifies what to do if the server jobs cannot run in the specified subsystem; for example if the subsystem is not active or if you want to prohibit certain users from using specific servers. Another example where the alternate action would be taken is when the necessary prestart jobs for the subsystem are not active. Possible values are **Reject**, and **Start in current subsystem**. When Reject is selected, the request will be rejected if it cannot run in the specified subsystem. When Start in current subsystem is selected, if the job cannot run in the specified subsystem, the job will attempt to run in the same subsystem that the server daemon is



running in. For the database and file servers this will typically result in the request being run in the QSERVER subsystem. For other servers, this will result in requests being sent to QSYSWRK. See the [Server table](#) for more information on server jobs and the subsystems in which they run, including server and daemon jobs.

- **Specific clients**

Select this option if you want to set a unique subsystem configuration for specific clients. When you add specific client configuration, a <Public> entry will be added to the end of the list. The <Public> entry applies to all clients not included by the specific client entries you have added.

There are different actions you can take to configure specific server subsystem clients.

- **Add** - Select **Add** to add a client to the list. From **Add Client** you can specify the subsystem configuration for a specific client or a group of clients.

**IPv4 or IPv6** - Select the type of IP version you want to add.

**Description** - Enter a text description of the client(s) that you are configuring.

**Client IP address or IP address range** - Specify whether you want to use an individual IP address for a single client or a range of IP addresses for a group of clients. IP address ranges cannot overlap for the selected server.

**Subnet mask** - The subnet mask specifies the subnet mask for this IP address. The subnet mask is a unique 32-bit integer that defines the part of the network where an interface attaches. The mask is expressed in the form xxx.xxx.xxx.xxx, where each field is the decimal representation of one byte (8 bits) of the mask. For example, the subnet mask with a hexadecimal representation is 'X'FFFFFF00' is expressed as 255.255.255.0.

**Note:** If the entry is intended to only impact one IP address, the proper subnet mask would be 255.255.255.255.

**Subsystem** - Specify the subsystem you want these client(s) to run in. You can type in the ten character name of the subsystem you want to use, if it is not listed. If None is selected, the client(s) will perform the **Alternate action**.

**Alternate action** - The **Alternate action** list specifies what to do if the server jobs cannot run in the specified subsystem; for example if the subsystem is not active or if you want to prohibit certain users from using specific servers. Another example where the alternate action would be taken is when the necessary prestart jobs for the subsystem are not active. Possible values are **Reject** and **Start in current subsystem**. When Reject is selected, the request will be rejected if it cannot run in the specified subsystem. When Start in current subsystem is selected, if the job cannot run in the specified subsystem, the job will attempt to run in the same subsystem that the server daemon is running in. For the database and file servers this will typically result in the request being run in the QSERVER subsystem. For other servers, this will result in requests being sent to QSYSWRK. See the [Server table](#) for more information on server jobs and the subsystems in which they run, including server and daemon jobs.

- **Edit** - Select the client record that you want to edit, and then select **Edit** to make changes to the selected client record.
- **Remove** - Select the client record that you want to remove, and then select **Remove** to delete the selected client record from the list. You cannot remove the <Public> entry at the end of the list.

After all selections have been made, click **OK** to accept and apply the specified changes.

### ***Routing server jobs by user profile***

Some of the servers provided by IBM i can be configured to route jobs doing work on behalf of a specific user profile to a defined subsystem.

The list of servers that allow subsystem routing by user can be found in the SQL [SET\\_SERVER\\_SBS\\_ROUTING](#) procedure. You can configure only these servers to route a user to a specific subsystem. You must use the SQL [SET\\_SERVER\\_SBS\\_ROUTING](#) procedure to associate a user or group profile with both the server name and a subsystem name. When configured, new incoming TCP/IP server

connections will be routed to the subsystem specified. If the server is also configured with IBM Navigator for i server subsystem configuration, the job will first be routed based on the IBM Navigator for i server subsystem configuration and then the SQL SET\_SERVER\_SBS\_ROUTING configuration will be checked and the job will be rerouted if the subsystem and the necessary prestarted jobs are active and available. See the [Routing server jobs to a subsystem](#) for more information on IBM Navigator for i server subsystem configuration. To configure DRDA connections for user XYZUSER to run in the subsystem 'MYSBS' just created in [Creating a server subsystem](#) follow these steps:

1. If you used CRTDUPOBJ to create a copy of QUSRWRK, you should already have a prestart job entry in the new subsystem (MYSBS) for DRDA. If not, add the QRWTSRVR prestart job entry as follows:

```
ADDPJE SBSDB(SBSLIB/MYSBS) PGM(QSYS/QRWTSRVR)INLJOBS(1) THRESHOLD(1)
ADLJOBS(2) MAXJOBS(200) JOBD(QGPL/QDFTSVR) CLS(QSYS/QSYSCLS20)
```

**Note:** See [Tuning prestart job entries](#) for more information on setting the Initial number of jobs, threshold and additional number of jobs parameters.

2. From RunSQL Scripts in IBM Navigator for i, or STRSQL from an IBM i command line, run the following SQL statement:

```
CALL QSYS2.SET_SERVER_SBS_ROUTING('XYZUSER', 'QRWTSRVR', 'MYSBS')
```

3. Start the subsystem:

```
STRSBS SBSLIB/MYSBS
```

Check to make sure the subsystem is active and the QRWTSRVR prestart jobs were started. New DRDA connections should now be routed to MYSBS for user XYZUSER.

4. To remove the route by user customization for user XYZUSER, call the same SQL procedure with *null* for the subsystem name. For example:

```
CALL QSYS2.SET_SERVER_SBS_ROUTING('XYZUSER', 'QRWTSRVR', null)
```

5. To view all route by user customizations for the server, call the following SQL procedure:

```
SELECT * FROM QSYS2.SERVER_SBS_ROUTING
```

## Creating a controlling subsystem

IBM supplies two complete controlling subsystem configurations: QBASE (the default controlling subsystem), and QCTL. Only one controlling subsystem can be active on the system at one time. Typically, the IBM supplied subsystem configurations should be sufficient for most business needs. However, you can create your own version of a controlling subsystem and configure it to more closely meet your company's unique needs.

Use the IBM-supplied controlling subsystem QBASE or QCTL as a model for creating your own controlling subsystem.

**Note:** If you create your own controlling subsystem, you should use a name other than QBASE or QCTL.

The subsystem description for the controlling subsystem should contain the following:

- A routing entry containing:
  - Either \*ANY or QCMDI as routing data
  - QSYS/QCMD as the program to be called
  - Class QSYS/QCTL or a user-defined class. (This is because a user, typically the system operator, must be able to enter commands to do such things as free up storage if the auxiliary storage threshold has been reached.)
- A workstation entry for the console with a type of \*SIGNON (\*SIGNON is a value for the AT parameter, specified on the Add Work Station Entry (ADDWSE) command.)

The \*SIGNON value indicates that the sign-on display is shown at the workstation when the subsystem is started. This requirement ensures that the subsystem has an interactive device for the entry of the system and subsystem level commands. The End System (ENDSYS) command ends the IBM i licensed program to a single session (or sign-on display) at the console in the controlling subsystem. A subsystem description that does not contain a workstation entry for the console cannot be started as a controlling subsystem.

- An entry for another workstation:

This provides an alternative source of controlling input. If a console problem is detected during an attended IPL and the If console problem occurs (QSCPFCONS) system value is set to 1, the IPL continues in unattended mode. Then, if the subsystem description for the controlling subsystem contains a workstation entry for another workstation, that alternate workstation can be used.

- A routing entry containing:
  - QSYS/QARDRIVE as the program to be called,
  - and QSYS/QCTL as the class

After you have created the controlling subsystem, change the Controlling subsystem/library (QCTLSBSD) system value as follows (assuming the description is named QGPL/QCTLA):

```
CHGSYSVAL SYSVAL(QCTLSBSD) VALUE('QCTLA QGPL')
```

The change becomes effective at the next IPL.

### **Related concepts**

#### The controlling subsystem

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 via the system console. It is identified in the Controlling subsystem/library (QCTLSBSD) system value.

### **Related information**

Experience Report: Restricted State

## **Placing the system in restricted state**

If all of the subsystems, including the controlling subsystem are ended, the system goes into a restricted condition. You can place the system in a restricted condition by using one of two commands from an interactive workstation.

**Command:** End Subsystem with the \*ALL parameter (ENDSBS SBS(\*ALL))

**Command:** End System (ENDSYS)

**Important:** The ENDSBS or ENDSYS command should be issued from an interactive job in the controlling subsystem, and only from a workstation whose entry in the controlling subsystem description specifies AT(\*SIGNON). The interactive job from which the command was issued remains active when the controlling subsystem goes into a restricted condition. If the job issuing the command is one of two jobs that are active at the workstation (using the System Request key or the TFRSECJOB command), neither of the jobs is forced to end. However, the controlling subsystem does not end for the restricted condition until you end one of the jobs. Suspending group jobs also prevents the controlling subsystem from ending (until the group jobs are ended).

When the system is in the restricted condition, most of the activity on the system has ended, and only one workstation is active. The system must be in this condition for commands such as Save System (SAVSYS) or Reclaim Storage (RCLSTG) to run.

Some programs for diagnosing equipment problems also require the system to be in a restricted condition. To end the restricted condition you must start the controlling subsystem again.

### **Related concepts**

#### The controlling subsystem

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 via the system console. It is identified in the Controlling subsystem/library (QCTLSBSD) system value.

#### **Related information**

[Experience Report: Restricted State](#)

## **Managing memory pools**

Making sure that jobs get enough memory to complete efficiently is important. If too much memory is given to subsystem A and not enough to subsystem B, jobs in subsystem B might begin to run poorly. The following information describes the various tasks that are involved in managing memory pools.

#### **Related concepts**

[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 your system, all main storage can be divided into logical allocations called memory pools. By default, the system manages the transfer of data and programs into memory pools.

#### **Related information**

[Managing system performance](#)

[Basic performance tuning](#)

[Applications for performance management](#)

[Experience report: The Performance Adjuster \(QPFRADJ\)](#)

## **Viewing memory pool information**

You can view information about the memory pools that are on your system by using IBM Navigator for i or the character-based interface.

#### **Related concepts**

[Memory pool allocation](#)

When you start a subsystem, the system attempts to allocate the user-defined storage pools that are defined in the subsystem description of the started subsystem.

[Memory pool activity level](#)

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. This allows for efficient use of system resources. The system manages the control of the activity level.

#### **Related information**

[Managing system performance](#)

[Basic performance tuning](#)

[Applications for performance management](#)

[Experience report: The Performance Adjuster \(QPFRADJ\)](#)

## ***IBM Navigator for i***

Expand **Work Management > All Tasks > Memory Pools > Active Memory Pools** or **Shared Memory Pools**.

The Active Memory Pools container displays both shared and private pools as long as they are active. The Shared Memory Pools container displays all of the shared pools regardless of their current status. Inactive private pools do not exist beyond the pool definition until they are activated by the subsystem. Thus they cannot be viewed using IBM Navigator for i.

## ***Character-based interface***

**Command:** Display Subsystem Description (DSPSBSD)

Use option 2 - Pool Definitions to view all of the private and shared pool definitions that exist in this subsystem definition.

**Command:** Work with Shared Pools (WRKSHRPOOL)

## Determining the number of subsystems using a memory pool

Subsystems are allocated a certain percentage of memory to run jobs. It is important to know how many different subsystems are pulling from the same memory pool. After you know how many subsystems are submitting jobs to a pool and how many jobs are running in a pool, you might want to reduce resource contention by adjusting the size and activity level of the pool.

### Related concepts

[Memory pool allocation](#)

When you start a subsystem, the system attempts to allocate the user-defined storage pools that are defined in the subsystem description of the started subsystem.

[Memory pool activity level](#)

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. This allows for efficient use of system resources. The system manages the control of the activity level.

### Related information

[Managing system performance](#)

[Basic performance tuning](#)

[Applications for performance management](#)

[Experience report: The Performance Adjuster \(QPFRADJ\)](#)

## *IBM Navigator for i*

To monitor the number of subsystems that are using a memory pool, follow the following instructions:

1. Expand **Work Management** > **Memory Pools** > **Active Memory Pools** or **Shared Memory Pools**.
2. Right-click the memory pool that you want to work with and click **Subsystems**.

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

## *Character-based interface*

**Command:** Work with Subsystems (WRKSBS)

This command displays a list of all of the subsystems and their corresponding pools.

## Determining the number of jobs in a memory pool

IBM Navigator for i provides you with a way to quickly display a list of jobs that are currently running in a memory pool.

To determine the number of jobs in a memory pool, follow the following instructions:

1. Expand **Work Management** > **All Tasks** > **Memory Pools** > **Active Memory Pools** or **Shared Memory Pools**.
2. Right-click the memory pool you want to use and click **Jobs**. A window 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.

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

### Related concepts

[Memory pool allocation](#)

When you start a subsystem, the system attempts to allocate the user-defined storage pools that are defined in the subsystem description of the started subsystem.

#### Memory pool activity level

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. This allows for efficient use of system resources. The system manages the control of the activity level.

#### **Related information**

Managing system performance

Basic performance tuning

Applications for performance management

Experience report: The Performance Adjuster (QPFRADJ)

## **Determining in which pool a single job is running**

If you have a job that is not performing as you expect you might want to check the memory pool in which the job is running. To determine in which pool a single job is running, use IBM Navigator for i or the character based interface.

After you have identified the pool in which the job is running, you can view memory pool information and determine if changes need to be made. For example, if too much paging occurs, possibly the memory pool needs to be larger. Another possibility for poor performance is that maybe too many other jobs are in the pool and you need to route this job to another pool.

#### **Related concepts**

#### Memory pool allocation

When you start a subsystem, the system attempts to allocate the user-defined storage pools that are defined in the subsystem description of the started subsystem.

#### Memory pool activity level

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. This allows for efficient use of system resources. The system manages the control of the activity level.

#### **Related information**

Managing system performance

Basic performance tuning

Applications for performance management

Experience report: The Performance Adjuster (QPFRADJ)

## ***IBM Navigator for i***

1. Expand **Work Management** > **Active Jobs** or **Server Jobs**, depending on the type of job you want to work with.
2. Find the job whose memory pool you want to view.
3. Right-click the **Job Name** and click **Properties**.
4. Click the **Resources** tab. The Job Properties - Resources window displays specific information about the job's memory pool.

#### *Character-based interface*

**Command:** Work with job (WRKJOB)

**Option 1:** Display Job Status Attributes

The Subsystem pool ID field contains the name of the pool defined for the subsystem in which the job is running. This field is blank for jobs that are not active at the time the display is requested. It is also blank for system jobs (type SYS), subsystem monitor jobs (type SBS) that do not run within a subsystem and batch immediate jobs (BCI) that are running in the Base memory pool.

**Command:** Work with active job (WRKACTJOB)

You can use the WRKACTJOB command to see the system pool ID for an active job.

## Managing tuning parameters for shared pools

To manage tuning parameters for shared pools, use IBM Navigator for i or character-based interface commands.

### Related concepts

[Pool numbering schemes](#)

Pools have two sets of numbering schemes: one is used within a subsystem and one is system-wide. The subsystem uses a set of numbers that refer to the pools it uses. Thus, when you create or change a subsystem description you can define one or more pools and label them 1, 2, 3, and so on. These are the designations of the subsystem pools, and they do not correspond to the pool numbers shown on the Work with System Status (WRKSYSSTS) display.

### Related information

[Managing system performance](#)

[Basic performance tuning](#)

[Applications for performance management](#)

[Experience report: The Performance Adjuster \(QPFRADJ\)](#)

## IBM Navigator for i

To access tuning parameters, use the following instructions:

1. Expand **Work Management > All Tasks > Memory Pools > Active Memory Pools or Shared Memory Pools**.
2. Right-click the pool that you want to tune and click **Properties**.
3. Click the **Tuning** tab.

From the Shared Properties - Tuning window you can manually adjust specific values such as pool allocation percentage, page faults per second, and priority.

## Character-based interface

**Command:** Work with Shared Pool (WRKSHRPOOL)

Select **Option 11 - Display tuning data** .

## Managing a pool's configuration

To change a pool's size, activity level or paging option, use IBM Navigator for i or character-based interface commands.

### Related concepts

[Pool numbering schemes](#)

Pools have two sets of numbering schemes: one is used within a subsystem and one is system-wide. The subsystem uses a set of numbers that refer to the pools it uses. Thus, when you create or change a subsystem description you can define one or more pools and label them 1, 2, 3, and so on. These are the designations of the subsystem pools, and they do not correspond to the pool numbers shown on the Work with System Status (WRKSYSSTS) display.

### Related information

[Managing system performance](#)

[Basic performance tuning](#)

[Applications for performance management](#)

[Experience report: The Performance Adjuster \(QPFRADJ\)](#)



## ***IBM Navigator for i***

To access a shared pool's configuration values, use the following instructions:

1. Expand **Work Management > All Tasks > Memory Pools > Active Memory Pools or Shared Memory Pools**.
2. Right-click the pool that you want to tune and click **Properties**.
3. Click the **Configuration** tab.

From the Shared Properties - Configuration window you can manually adjust specific values such as a pool's size, activity level or paging option.

### ***Character-based interface***

**Command:** Work with Shared Pool (**WRKSHRPOOL**)

### **Changing memory pool size**

The size of a memory pool directly affects the amount of work that a subsystem can process. The more memory a subsystem has, the more work it can potentially complete. It is important that you monitor your system carefully before you start changing the parameters of your memory pools. You also want to periodically recheck these levels, as some readjustment might need to be done.

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 might be changed automatically by the tuner.

Turn the system tuner off by changing the Automatically adjust memory pools and activity levels (QPFRADJ) system value to 0. (0 = No adjustment)

#### **Related concepts**

[Pool numbering schemes](#)

Pools have two sets of numbering schemes: one is used within a subsystem and one is system-wide.

The subsystem uses a set of numbers that refer to the pools it uses. Thus, when you create or change a subsystem description you can define one or more pools and label them 1, 2, 3, and so on. These are the designations of the subsystem pools, and they do not correspond to the pool numbers shown on the Work with System Status (WRKSYSSTS) display.

#### **Related information**

[Managing system performance](#)

[Basic performance tuning](#)

[Applications for performance management](#)

[Experience report: The Performance Adjuster \(QPFRADJ\)](#)

## ***IBM Navigator for i***

1. Expand **Work Management > All Tasks > Memory Pools > Active Memory Pools or Shared Memory Pools**.
2. Right-click the memory pool that you want to work in (for example, Interactive) and click **Properties**. The **Memory Pool Properties** window appears.
3. 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 the pool can use. The number you put here should reflect the amount of memory that you think the pool needs to support the subsystems it services.

**Note:** The Base pool is the only memory pool that does not have a defined amount of memory. Instead 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 might 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 is not allocated to anything. This non-allocated 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.

### ***Character-based interface***

**Command:** Change System Value (CHGSYSVAL )

**Example:** The following changes the size of the Machine pool.

```
CHGSYSVAL QMCHPOOL 'new-size-in-KB'
```

This corresponds to pool 1 on the WRKSYSTS display.

**Example:** The following changes the minimum size of the base pool.

```
CHGSYSVAL QBASPOOL 'new-minimum-size-in-KB'
```

This corresponds to pool 2 on the WRKSYSSTS display.

**Note:** The QBASPOOL system value only controls the minimum size of the base pool. The Base pool contains all of the storage that is not allocated to other pools.

### ***Change the size of a shared pool***

**Command:** Change Shared Storage Pool (CHGSHRPOOL)

The changes to shared pools take effect immediately if the shared pool is active and sufficient storage is available.

**Command:** Work with Shared Storage Pools (WRKSHRPOOL)

This command gives you access to the names and status information of shared pools. By using the menu options you can change values for pool size and maximum activity levels.

## **Creating a private memory pool**

Private memory pools (also known as user-defined memory pools) can be used by IBM-supplied subsystems or by user-defined subsystems. You can define up to a maximum of 10 memory pool definitions for a subsystem. You create a private memory pool in the subsystem description.

To create a private memory pool, use the character-based interface.

**Command:** Create Subsystem Description (CRTSBSD) POOLS parameter.

**Command:** Change Subsystem Description (CHGSBSD) POOLS parameter.

**Note:** Although each subsystem description can have as many as 10 user-defined memory pools, there is an operation limitation of no more than 64 memory pools that can be active at any time. (This includes the Base memory pool and the Machine memory pool.) If the maximum allocation limit is reached before all of the memory pools for a subsystem are allocated, the Base pool is used for any routing steps that still require a memory pool.

### **Related concepts**

[Types of memory pools](#)

On the your system, all main storage can be divided into logical allocations called *memory pools*. All memory pools in a system are either private or shared. There are private memory pools, shared memory pools, and special shared memory pools. As many as 64 memory pools, in any combination of private and shared pools, can be active at the same time.

### **Related information**

[Managing system performance](#)

[Basic performance tuning](#)

[Applications for performance management](#)

[Experience report: The Performance Adjuster \(QPFRADJ\)](#)

[Performance system values: Machine memory pool size](#)

[Performance system values: Base memory pool minimum size](#)

[Performance system values: Base memory pool maximum eligible threads](#)

## Managing job queues

As you manage the work on your system, you might find it necessary to manipulate jobs that are waiting in a job queue. Perhaps someone needs a job run immediately and the job is sitting in a queue at a low priority. Or maybe you need to perform some maintenance on a subsystem and want to move all of the jobs to a queue that is not associated with that particular subsystem.

The following information describes how to accomplish these types of management tasks.

### Related concepts

#### [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.

## Assigning the job queue to the subsystem

To assign a job queue entry to a subsystem description, use the character-based interface.

**Command:** Add Job Queue Entry (ADDJOBQE)

The parameters on this command specify:

- The number of jobs that can be active at the same time on this job queue (MAXACT)
- In what order the subsystem handles work from this job queue (SEQNBR)
- How many jobs can be active at one time for each of the nine levels of priority (MAXPTYn) (n=1 through 9)

**Example:** The following example adds a job queue entry for the JOBQA job queue in the TEST subsystem description. There is no maximum number of jobs that can be active on this job queue and the work is processed with a sequence number of five.

```
ADDJOBQE SBSD(TEST) JOBQ(LIBA/JOBQA) MAXACT(*NOMAX) SEQNBR(5)
```

### Related concepts

#### [How a job queue works](#)

Job queues are allocated by a subsystem via the job queue entry. Jobs can be placed on a job queue even if the subsystem has not been started. When the subsystem is started, it processes the jobs on the queue.

### Related tasks

#### [Removing job queue entries](#)

You can remove job queue entries from a subsystem description by using the character-based interface. Jobs on the job queue remain on the queue when the job queue entry is removed from the subsystem description. A job queue entry cannot be removed if any currently active jobs were started from the job queue.

## *How a subsystem handles several job queues*

To illustrate how a subsystem handles several job queues, consider this scenario.

### **Job Queue A (SEQNBR=10)**

Job 1

Job 2

Job 3

### **Job Queue B (SEQNBR=20)**

Job 4

Job 5

Job 6

### **Job Queue C (SEQNBR=30)**

Job 7

Job 8

Job 9

Each job queue entry in this scenario is specified as MAXACT(\*NOMAX). The subsystem first selects jobs from job queue **A** because the job queue entry has the lowest sequence number. If the maximum number of jobs in the subsystem is 3 (MAXJOBS(3) parameter on the Create Subsystem Description (CRTSBSD) command), it can select all the jobs from job queue **A** to be active at the same time.

When any of the three jobs is completed, the activity level is no longer at the maximum; therefore a new job is selected from job queue **B** because it has the next lowest sequence number (assuming no new jobs have been added to job queue **A**). Because each job queue entry specifies MAXACT(\*NOMAX), the MAXACT value does not prevent jobs from being started. Had each job queue entry specified MAXACT(1), then jobs 1, 4, and 7 would have been started. Had job queue entry **A** been specified as MAXACT(2), then jobs 1, 2, and 4 would have been started.

### **Related concepts**

#### How a job queue works

Job queues are allocated by a subsystem via the job queue entry. Jobs can be placed on a job queue even if the subsystem has not been started. When the subsystem is started, it processes the jobs on the queue.

## **Changing the number of jobs running simultaneously in a job queue**

The QBASE subsystem is shipped with a job queue entry for the QBATCH job queue. This entry only allows one batch job to run at a time. If you want more than one batch job from that job queue to run simultaneously then you need to change the job queue entry.

To change the number of jobs running simultaneously from a job queue, use the character-based interface.

**Command:** Change Job Queue Entry (CHGJOBQE)

**Example:** The following command allows two batch jobs from the QBATCH job queue to run at the same time in the QBASE subsystem. (This command can be issued at any time and takes effect immediately.)

```
CHGJOBQE SBS(D(QBASE) JOBQ(QBATCH) MAXACT(2)
```

### **Related concepts**

#### How jobs are taken from multiple job queues

A subsystem processes jobs from a job queue based on sequence number. A subsystem can have more than one job queue entry and can therefore allocate more than one job queue.

#### How jobs are taken from a job queue

Different factors determine how the jobs are selected from a job queue and started.

#### Job queue entry

A job queue entry identifies a job queue from which jobs are selected for running in the subsystem. There are five parameters in the job queue entry that control how the job queue should be handled.

## **Clearing a job queue**

When you clear a job queue, every job on the queue is deleted. This includes any jobs that are in the hold state. You can use IBM Navigator for i or the character-based interface to clear a job queue. Jobs that are running are not affected because they are considered active jobs and are no longer on the queue.

### **Related concepts**

#### 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.

## ***IBM Navigator for i***

To clear a job queue, follow these steps:

1. Expand **Work Management > All Tasks > Job Queues > Active Job Queues or All Job Queues**.
2. Right-click the job queue and click **Clear**.

The Confirm Clear window appears where you can specify if you want a job log produced when the queue clears.

## ***Character-based interface***

**Command:** Clear Job Queue (CLRJOBQ)

**Example:** This command removes all jobs currently in the IBM-supplied job queue, QBATCH. Any job currently being read in is not affected.

```
CLRJOBQ JOBQ(QGPL/QBATCH)
```

## **Creating job queues**

To create a job queue, use the character-based interface.

**Command:** Create Job Queue ( CRTJOBQ )

**Example:** The following example creates a job queue called JOBQA in the LIBA library:

```
CRTJOBQ JOBQ(LIBA/JOBQA) TEXT('test job queue')
```

After you create a job queue, it must be assigned to a subsystem before any jobs can be run. To assign a job queue to a subsystem, add a job queue entry to the subsystem description.

### **Related concepts**

#### 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.

#### How a job queue works

Job queues are allocated by a subsystem via the job queue entry. Jobs can be placed on a job queue even if the subsystem has not been started. When the subsystem is started, it processes the jobs on the queue.

## **Deleting a job queue**

To delete a job queue, use the character-based interface.

Restrictions:

- The job queue being deleted cannot contain any entries. All jobs on the queue must be completed, deleted, or moved to a different job queue.
- A subsystem cannot be active to the job queue.

There is more than one way to delete a job queue. Although two methods are listed here, the WRKJOBQ command is the recommended method because it shows the job count and status.

**Command:** Work with Job Queue (WRKJOBQ)

If the number of jobs is 0, then you can use option 4=Delete to delete the job queue from the library.

Use the DLTJOBQ with automated scripts and clean up environments. Be careful when using this method because the default behavior of this command is to search the library list and delete the first job queue that matches the specified name. If you have two job queues of the same name in different libraries, you might end up deleting the wrong one. You can override this behavior by specifying a specific library.

**Command:** Delete Job Queue (DLTJOBQ)

**Example:** This command deletes the job queue SPECIALJQ in library SPECIALLIB.

```
DLTJOBQ JOBQ(SPECIALLIB/SPECIALJQ)
```

### Related concepts

#### 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.

## Determining which subsystem has a job queue allocated

You can determine which subsystem has allocated the job queue using the IBM Navigator for i interface or the character-based interface. This is useful when you find it necessary to delete the job queue since you cannot delete a job queue to which a subsystem is active.

### Related concepts

#### How a job queue works

Job queues are allocated by a subsystem via the job queue entry. Jobs can be placed on a job queue even if the subsystem has not been started. When the subsystem is started, it processes the jobs on the queue.

### *IBM Navigator for i*

To see which subsystem has allocated the job queue, follow the following instructions:

1. From IBM Navigator for i, expand **Work Management > All Tasks > Job Queues > All Job Queues**.
2. Locate the job queue in the right pane of the IBM Navigator for i interface.  
The subsystem which has allocated the job queue is displayed in the Subsystem column.  
(If you do not see the Subsystem column, add it to the display. Click **Actions > Columns**.)
3. Or you can right-click the job queue and click **Properties**. The subsystem is listed on the General page of the Job Queue Properties window.

### *Character-based interface*

**Command:** WRKJOBQ JOBQ(LIBA/JOBQA) where JOBQA is the name of the job queue

1. Type in the command WRKJOBQ JOBQ(LIBA/JOBQA).  
The Work with Job Queue display appears. The subsystem description function key appears in the function key area of the display when the job queue is allocated to a system.
2. Press the subsystem description function key.  
The Work with Subsystem Descriptions display appears showing the subsystem to which the job queue is allocated.

## Holding a job queue

When you place a job queue on hold you prevent the processing of all of the jobs that are currently waiting on the job queue. Placing a job queue on hold has no effect on jobs that are running. Additional jobs can be placed on the job queue while it is held, but they are not processed.

To place a job queue on hold, you can use IBM Navigator for i or the character-based interface.

### Related concepts

#### 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.

### ***IBM Navigator for i***

Expand **My Connections > connection > Work Management > Job Queues > Active Job Queues > Right-click the queue > Hold.**

*Character-based interface*

**Command:** Hold Job Queue (HLDJOBQ)

In this example, the job queue QBATCH is placed on hold. All jobs that are not running at the time that the command is issued are held until the queue is released or cleared.

```
HLDJOBQ  JOBQ(QBATCH)
```

### **Releasing a job queue**

When you release a job queue, all of the jobs that were placed on hold as a result of placing the job queue on hold are also released. If an individual job was placed on hold before the job queue was held, then the job is not released.

To release a job queue, use IBM Navigator for i or the character-based interface.

#### **Related concepts**

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.

### ***IBM Navigator for i***

Expand **Work Management > All Tasks > Job Queues > All Job Queues > Right-click the queue > Release.**

*Character-based interface*

**Command:** Release Job Queue (RLSJOBQ)

This example releases the job queue QBATCH.

```
RLSJOBQ  JOBQ(QBATCH)
```

### **Moving a job to a different job queue**

There are many reasons why you might want to move a job to another queue. For example, sometimes jobs become backlogged in the queue because of a long running job. Perhaps the job's scheduled run time conflicts with a new job that has a higher priority. One way to manage this situation is to move the waiting jobs to another queue that is not as busy.

You can use either the IBM Navigator for i interface or the character-based interface to move a job from one queue to another.

#### **Related concepts**

How jobs are taken from multiple job queues

A subsystem processes jobs from a job queue based on sequence number. A subsystem can have more than one job queue entry and can therefore allocate more than one job queue.

How jobs are taken from a job queue

Different factors determine how the jobs are selected from a job queue and started.

### ***IBM Navigator for i***

1. Expand **Work Management > All Tasks > Job Queues > All Job Queues**.
2. Right-click the job queue that contains the job and select **Jobs**.
3. Right-click the job that you want to move.

The Move window opens where you can specify the target queue.

- Jobs that are waiting to run are moved to the same relative position on the target queue (for example, jobs with a job queue priority 3 are moved after any other priority 3 jobs that are waiting to run on the target queue).
- Jobs that are held remain held and are placed in the same relative position on the target queue (for example, held jobs with job queue priority 3 are moved after any other priority 3 held jobs on the target queue).
- Jobs that are scheduled to run are moved to the target queue and their scheduled times remain unchanged.

#### *Character-based interface*

**Command:** Change Job (CHGJOB)

**Example:** The following example moves job JOBA to job queue JOBQB.

```
CHGJOB JOB(JOBA) JOBQ(LIBA/JOBQB)
```

### **Placing a job on the job queue**

Jobs are placed on the job queue by either moving an existing job from one queue to another, or by submitting a new job. Use IBM Navigator for i to move jobs between queues. Use the character-based interface to submit a new job.

#### **Related concepts**

[How jobs are taken from multiple job queues](#)

A subsystem processes jobs from a job queue based on sequence number. A subsystem can have more than one job queue entry and can therefore allocate more than one job queue.

[How jobs are taken from a job queue](#)

Different factors determine how the jobs are selected from a job queue and started.

### ***IBM Navigator for i***

To use the IBM Navigator for i interface, the job must already exist in another job queue. Then you can move the job from one queue to another queue. (To place a new job on a job queue, use the command line interface.)

1. Expand **Work Management > All Tasks > Job Queues > All Job Queues**.
2. Right-click the job queue that contains the job and select **Jobs**.
3. Right-click the job that you want to move.

The Move window opens where you can specify the destination queue.

#### ***Character-based interface***

The following is a list of character-based interface methods for placing a new job on the new job queue.

- **Submit Job (SBMJOB):** Allows a job that is running to submit another job to a job queue to be run later as a batch job. Only one element of request data can be placed on the new job's message queue. The request data can be a CL command if the routing entry used for the job specifies a CL command processing program (such as the IBM-supplied QCMD program).
- **Add Job Schedule Entry (ADDJOBSCDE):** Automatically the system submits a job to the job queue at the time and date specified in the job schedule entry.

- **Submit Database Jobs (SBMDBJOB):** Submits jobs to job queues so they can be run as batch jobs. The input stream is read either from a physical database file or from a logical database file that has a single-record format. This command allows you to specify the name of this database file and its member, the name of the job queue to be used, and to decide whether jobs being submitted can be displayed by the Work with Submitted Jobs (WRKSBMJOB) command.
- **Start Database Reader (STRDBRDR):** Read a batch input stream from a database and place one or more jobs on job queues.
- **Transfer Job (TFRJOB):** Move the current job to another job queue in an active subsystem.
- **Transfer Batch Job (TFRBCHJOB):** Move the current job to another job queue.

## Searching all job queues for a specific job

You can use either the IBM Navigator for i or the character-based interface to search job queues for a specific job.

### *IBM Navigator for i*

1. Expand **Basic Operations > User Jobs > Actions > Include.**
2. Use the Jobs-Include window to narrow down the number of jobs that are displayed. Make sure that the **Job queue** field is set to All.
3. When you click **OK** all of the jobs that meet your criteria are displayed.

### *Character-based interface*

**Command:** Work with Job Queues ( WRKJOBQ)

**Example:** The following example creates a list of all jobs on the JOBQA job queue.

```
WRKJOBQ JOBQ(LIBA/JOBQA)
```

### *Find a job when you do not know the name of the job queue*

If you do not know the name of the job queue, follow the following instructions:

1. Enter the command without the JOBQ parameter.  
The Work with All Job Queues window appears with a list of all job queues to which you are authorized.
2. Scan this list until you see the name of a job queue that may contain the job that you are trying to find.

After you have found a job in a job queue, you can look at that job by entering the work with option for the job that you want to see. The Work with Job display appears. This display provides several options for viewing all of the information available for the job that you selected.

If you know what job you are looking for, the following command can take you directly to the job display.

```
WRKJOB JOB(number/user/name) OPTION(*DFNA)
```

If you're not sure what job you're looking for, Work with Submitted Jobs (WRKSBMJOB) or Work with User Jobs (WRKUSRJOB) can help.

## Specifying the priority for the job queue

To specify the order in which the job queues are processed by the subsystem, use the character-based interface.

**Command:** Add Job Queue Entry ( ADDJOBQE)

The parameters on this command specify:

- Number of jobs that can be active at the same time on this job queue (MAXACT)
- In what order the subsystem handles work from this job queue (SEQNBR)



- How many jobs can be active at one time for each of nine levels of priority (MAXPTYn) (n=1 through 9)

## Managing 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.

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.

### Related concepts

#### 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.

## Creating an output queue

The Create Output Queue (CRTOUTQ) command creates a new output queue for spooled files. An entry is placed on the output queue for each spooled file. The order in which the files are written to the output device is determined by the output priority of the spooled file and the value specified on the Order of files on queue prompt (SEQ parameter). Use the character-based interface to create an output queue.

**Command:** CRTOUTQ (Create Output Queue)

**Example:** This command creates an output queue named DEPTAPRT and puts it in the current library. Because AUT(\*EXCLUDE) is specified and OPRCTL(\*YES) is assumed, the output queue can be used and controlled only by the user who created the queue and users who have job control authority or spooled control authority. Because SEQ(\*FIFO) is specified, spooled files are placed in first-in first-out order on the queue. If users in Department A are authorized to use this output queue, the Grant Object Authority (GRTOBJAUT) command must be used to grant them necessary authority. Data contained in files on this queue can be displayed only by users who own the files, by the owner of the queue, by users with job control authority, or by users with spool control authority. By default, no job separator is printed at the beginning of the output for each job.

```
CRTOUTQ  OUTQ(DEPTAPRT) AUT(*EXCLUDE) SEQ(*FIFO)
          TEXT('SPECIAL PRINTER FILES FOR DEPTA')
```

Example: The following is another example of how you can create an output queue.

```
CRTOUTQ  OUTQ(QGPL/JONES) +
          TEXT('Output queue for Mike Jones')
```

## Assigning the output queue to a job or job description

Before you can use a newly created output queue, you need to assign it to a job or to a job description. You can assign the output queue by using IBM Navigator for i or the character-based interface.

### IBM Navigator for i

To use IBM Navigator for i to assign the output queue to a job, follow these steps:

1. Expand **Work Management > Active Jobs**.
2. Right-click a job and click **Printer Output**.

### Character-based interface

You can also change the job description to use the new output queue. Thus, all of the jobs that use the job description use the new output queue. Use the character-based interface to assign an output queue to a job description.

**Command:** Change Job Description (CHGJOB)

The following example changes the job description AMJOBS so that the output queue QPRINT is used.

```
CHGJOB JOB(AMJOBS/AMJOBS) OUTQ(*LIBL/QPRINT)
```

## Accessing 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 IBM Navigator for i through Basic Operations or through Work Management.

### *IBM Navigator for i*

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

1. Expand **Basic Operations**.
2. Right-click the job for which you want to display printer output and click **Printer Output**. The Printer Output window appears.

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

1. Expand **Work Management > Output Queues**.
2. Select the output queue with which you want to display printer output (for example, Qprint2). The printer output within the output queue appears.

### *Character-based interface*

**Command:** Work with Output Queue (WRKOUTQ <output queue name>)

**Command:** Work with Spooled Files (WRKSPLF JOB(qualified job name))

## Clearing output queues

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

### *IBM Navigator for i*

To clear an output queue, follow these steps:

1. Expand **Work Management > Output Queues**.
2. Right-click the output queue that you want to clear, and click **Clear**.

### *Character-based interface*

**Command:** Clear Output Queue (CLROUTQ)

This command removes the entries for all spooled files from the output queue, QPRINT, that are waiting to be printed or are being held. The entries for the file currently being printed and files still receiving data from programs that are currently running are not affected.

```
CLROUTQ OUTQ(QPRINT)
```

## Deleting an output queue

You can use the character-based interface to delete an output queue.

Before an output queue can be deleted, it must meet the following requirements.

The output queue being deleted cannot contain any entries. The output for each file must be printed, deleted, or moved to a different output queue. A subsystem cannot be active. The queue cannot be in use by a spooling writer. The queue cannot be deleted if it has been created by the system for a specific printer.

**Command:** Delete Output Queue (DLTOUTQ)

This command deletes the output queue PUNCH2 from the system.

```
DLTOUTQ  OUTQ(PUNCH2)
```

## Viewing output queues on the system

Output queues determine the order in which printer output is sent to the printer device. You can view output queues by using IBM Navigator for i.

To view output queues on the system, use the following instructions:

1. From IBM Navigator for i, expand **Work Management**.
2. Click **Output Queues**.

From IBM Navigator for i, you can customize the list of output queues you are viewing by using the Include window. The Include window allows you to put limitations on what is displayed. For example, you can run Include to display only certain output queues.

To use the include function, click **Actions > Include**.

## Managing job logs

Most jobs on your system have a job log associated with it. Job logs tell you 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 you a good idea of how the job cycle is running.

The following information discusses the various tasks that you can perform when working with job logs.

### Related concepts

#### Job logs

A job log contains information related to requests entered for a job. A job log has two forms, a pending form and a spooled form.

## Managing the job log server

The QSYSWRK subsystem controls the job log server. However, there are some tasks that you can perform to customize or manage the job log server.

### Related concepts

#### Job logs

A job log contains information related to requests entered for a job. A job log has two forms, a pending form and a spooled form.

### Reconfiguring the job log server

As shipped, the job log server runs in QSYSWRK. QSYSWRK is continuously active. To enhance performance, you might want to reconfigure your job log server to run in a different subsystem.

To reconfigure the job log server to run in a different subsystem, use the character-based interface and follow these steps.

1. Add a routing entry identical to the one from QSYSWRK to your subsystem description.  
This is the routing entry Seq Nbr 500, Program QWCJLSVR, Library QSYS, Compare Value 'QJOBLOGSVR', Start Pos 1.
2. Change the job queue specified in the QJOBLOGSVR job description to a job queue that is present on your subsystem.
3. Add the QJOBLOGAJ autostart job entry (along with a routing entry, if needed) to your subsystem. This causes the job log server to automatically start when your subsystem starts.
  - Or if you prefer, you can replace the autostart job entry with a call to the STRLOGSVR command in the startup program.

4. Remove the QJOBLOGAJ autostart job entry from QSYSWRK.

As another example of reconfiguring the job log server, you can use the Change Class (CHGCLS) command to change the Run priority (RUNPTY) specified in the QJOBLOGSVR class (in library QSYS).

```
CHGCLS CLS(QSYS/QJOBLOGSVR) RUNPTY(50)
```

### Related concepts

#### Job log server

Typically the job log server writes a job's job log to a spooled file. You can route the job log to a printer or to an outfile, (if specified to do so by using the QMHCTLJL, Control job log API), however this is not the recommended method for producing job logs.

### **Ending the job log server**

The End Job Log Server (ENDLOGSVR) command is used to end the job log server(s). The job log server writes job logs for jobs that are in a job log pending state. If more than one job log server job is active at the time this command is issued, all of the job log server jobs are ended.

You must have job control (\*JOBCTL) special authority to use this command.

**Important:** If you only want to stop the production of a particular job log because, for example, it is very long or consumes too many resources, see the related topic **Stop production of a particular job log**.

When using the ENDLOGSVR command, you can specify whether you want the server to end immediately (not recommended) or in a controlled manner.

### Related concepts

#### Job log server

Typically the job log server writes a job's job log to a spooled file. You can route the job log to a printer or to an outfile, (if specified to do so by using the QMHCTLJL, Control job log API), however this is not the recommended method for producing job logs.

### Related tasks

#### Stopping production of a specific job log

If you only want to stop the production of a particular job log, do not use the End Job Log Server (ENDLOGSVR) command. The ENDLOGSVR command ends all job log servers which results in stopping the production of all job logs.

#### Deleting job log output files

Job logs are removed from the system when a job completes normally, or when the Remove Pending Job Log (QWTRMVJL) API or the End Job (ENDJOB) command is issued. Additionally if "clear incomplete job logs" is specified on the IPL, all of the jobs in job log pending are removed from the system during an IPL. Any remaining job log output files can be found under **Basic Operations > Printer Output**.

#### *IBM Navigator for i*

1. Expand **System > Run Command**.
2. In Command to run: type ENDLOGSVR.
3. Click **Prompt**
4. The End Job Log Server window displays to help you specify the parameters of this command. Complete the window and click **OK**.  
The window closes and you are returned to the Run Command window.
5. You can now click **Run Command** to run the command immediately.

#### *Character-based interface*

**Command:** End Job Log Server (ENDLOGSVR)

## Starting the job log server

By default, the job log server automatically starts when the QSYSWRK subsystem starts. You can manually start a job log server by using the Start Job Log Server (STRLOGSVR) command.

When you use the STRLOGSVR command, you can specify the number of additional job log servers that you want to start, or you can let the system calculate the needed number for you. If the number of servers requested exceeds the maximum active allowed, only the difference between the maximum and current number of active servers will be started. The maximum number of job log servers that can be active on a job queue at one time is 30.

### Related concepts

#### Job log server

Typically the job log server writes a job's job log to a spooled file. You can route the job log to a printer or to an outfile, (if specified to do so by using the QMHCTLJL, Control job log API), however this is not the recommended method for producing job logs.

## IBM Navigator for i

1. Expand **System > Run Command**.
2. In **Command to run:** field type STRLOGSVR.
3. Click **Prompt**.
4. The Start Job Log Server window displays to help you specify the parameters of this command. Complete the window and click **OK**.  
The window closes and you are returned to the Run Command window.
5. You can now click **Run Command** to run the command immediately.

## Character-based interface

Command: **Start Log Server** (STRLOGSVR)

## How to display job logs

You can see a job log from any place within work management that you access jobs, such as through the Subsystem area or the Memory Pool area. You can use IBM Navigator for i or the character-based interface to display job logs.

### Related tasks

#### Producing printer output from job log pending

Jobs that do not have the IBM Navigator for i **Job Properties - Job Log** setting, **Produce a job log** field selected do not produce job logs. Instead the job log is in job log pending. To produce printer output from a job log that is in job log pending, use the character-based interface.

#### Deleting job log output files

Job logs are removed from the system when a job completes normally, or when the Remove Pending Job Log (QWTRMVJL) API or the End Job (ENDJOB) command is issued. Additionally if "clear incomplete job logs" is specified on the IPL, all of the jobs in job log pending are removed from the system during an IPL. Any remaining job log output files can be found under **Basic Operations > Printer Output**.

#### What to do when the job log does not display

In IBM Navigator for i, to find and display a job log, whether a batch job or an interactive job, right-click the job and then click **Job log** from the menu. However, depending upon the status of your job or how the job log values were set in the job description, your job log may be in the output queue, or it may be in a job log pending status or it may not be available.

#### Stopping a subsystem

You can use IBM Navigator for i or the character-based interface 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.

### Related information

#### Manage printing

## **IBM Navigator for i**

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

1. Expand **Work Management > Active Jobs** or **Server Jobs**.
2. Right-click a job (for example, Qbatch) and click **Job Log**. For more information, refer to the help in the Job Log window.

To view more details of a message, right-click a message and click **Properties**. The Message Properties window provides detailed message information. This window shows the details of the message as well as the message help. The detailed message help gives you information to solve a problem.

The following list describes additional ways to access job logs:

- **Basic Operations > Printers**
- **Basic Operations > User Jobs > Right-click a job > Printer Output**
- **Work Management > Active Jobs > Right-click a job > Printer Output**
- **Work Management > Output Queues**
- **Users and Groups > Users > Right-click a user > User Objects > Printer Output**

### *Character-based interface*

The way to display a job log depends on the status of the job.

- The **Work with Job Logs (WRKJOBLOG)** command can be used to display pending job logs for completed jobs, all job log spooled files, or both. For example, to display the list of pending job logs for all jobs that have ended use the following command:

```
WRKJOBLOG JOBLOGSTT(*PENDING)
```

- If the job is still active (batch or interactive jobs) or is on a job queue and has not yet started, use the **Display Job Log (DSPJOBLOG)** command. For example, to display the job log of the interactive job for user JSMITH at display station WS1, use the following command:

```
DSPJOBLOG JOB(nnnnnn/JSMITH/WS1)
```

where nnnnnn is the job number.

- If the job has ended and the job log is not yet printed, use the **Display Spooled File (DSPSPLF)** command. For example, to display the job logs for job number 001293 associated with user FRED at display station WS3, use the following command:

```
DSPSPLF FILE(QPJOBLOG) JOB(001293/FRED/WS3)
```

If you do not have enough information to use the above commands, the Work with User Jobs (WRKUSRJOB) command or the Work with Submitted Jobs (WRKSBMJOB) command might be helpful.

## **What to do when the job log does not display**

In IBM Navigator for i, to find and display a job log, whether a batch job or an interactive job, right-click the job and then click **Job log** from the menu. However, depending upon the status of your job or how the job log values were set in the job description, your job log may be in the output queue, or it may be in a job log pending status or it may not be available.

The following are some steps to take if the Job log menu option is unavailable for your job.

**Tip:** Set the column display for Active jobs (or Server jobs) to include the Status. This makes it easier to quickly determine where to look for your job log.

To access a job log: **Work Management > Active Jobs or Server Jobs > Right-click the job and select Job log**.

If the Job log menu option is not available or if you get an error message stating that the system is unable to retrieve the job log, consider the following:

1. Check the status of the job.

Option	Description
<b>Running</b>	Check the Job Properties - Job Log window and make sure the <b>Produce a job log</b> box is checked. If it is not checked, then no job log was produced.
<b>Ended</b>	This job did not end in a normal manner. It might be due to an error or user intervention. Right click the job and then click <b>Printer Output</b> . If you do not see your job log there, check the Job Properties - Job Log window and make sure the check box, Produce printer output for job log is selected.
<b>Completed - Printer output is available</b>	This job ended normally. Right click the job and then click <b>Printer Output</b> . If you do not see your job log there, check the Job Properties - Job Log window and make sure that the field <b>Create printer output for job log if job ends normally</b> was checked.
<b>Completed - Job log pending</b>	The job log is not produced. The job log remains in pending until removed. You need to use the Display Job Log (DSPJOBLOG) command to view the pending job log.

2. The job log may have been spooled to an output queue and has printed, in which case the log has been removed from the system.
3. Another possibility is that the job log was deleted by another user.

### Related tasks

#### How to display job logs

You can see a job log from any place within work management that you access jobs, such as through the Subsystem area or the Memory Pool area. You can use IBM Navigator for i or the character-based interface to display job logs.

## Specifying the output queue for a job log

By default the printer file that is used to spool a job log is QPJOBLOG. You can have multiple QPJOBLOG printer files on your system. In QSYS the output queue that the OUTQ attribute uses is QEZJOBLOG, in library QUSRSYS. When the system creates a job log, it looks for the printer file QPJOBLOG in the job's library list. The first one that is found is the one that it uses. You use the character-based interface to adjust these settings.

1. Change the printer file QPJOBLOG OUTQ attribute to \*JOB.
  - a) **Command:** Change Printer File CHGPRTF FILE(QPJOBLOG) OUTQ(\*JOB)
2. Change the job's OUTQ attribute to the output queue that you want.
 

You can do this by using the character-based interface or IBM Navigator for i.

  - a) Command: Change Job CHGJOB OUTQ(MYLIB/MYOUTQ)
  - b) IBM Navigator for i: **Work Management > Active Jobs > Right-click a job and select Properties > Printer Output**

### Related information

[Controlling printing to output queue or printer](#)

## Stopping production of a specific job log

If you only want to stop the production of a particular job log, do not use the End Job Log Server (ENDLOGSVR) command. The ENDLOGSVR command ends all job log servers which results in stopping the production of all job logs.

Instead, use the following procedure to stop production of a specific job log.

1. From IBM Navigator for i, right-click the job that you want to stop the job log production and click **Properties**.  
(**Work Management > Active Jobs or Server Jobs**)
2. Click the **Job Log** tab.
3. Uncheck the **Produce a job log** box and click **OK**.

Production of the job log will cease and the job log will be in a job log pending status.

#### **Related concepts**

##### How job logs are created

The job logs are available when needed, but no work is done to produce job logs for which there is no need.

#### **Related tasks**

##### Ending the job log server

The End Job Log Server (ENDLOGSVR) command is used to end the job log server(s). The job log server writes job logs for jobs that are in a job log pending state. If more than one job log server job is active at the time this command is issued, all of the job log server jobs are ended.

##### Deleting job log output files

Job logs are removed from the system when a job completes normally, or when the Remove Pending Job Log (QWTRMVJL) API or the End Job (ENDJOB) command is issued. Additionally if "clear incomplete job logs" is specified on the IPL, all of the jobs in job log pending are removed from the system during an IPL. Any remaining job log output files can be found under **Basic Operations > Printer Output**.

##### Controlling batch job log information

For your batch applications, you may want to change the amount of information logged. The log level (LOG(40 \*NOLIST)) specified in the job description for the IBM-supplied subsystem QBATCH supplies a complete log if the job abnormally ends. If the job completes normally, no job log is produced.

## **Preventing the production of a job log**

Preventing the production of a job log is useful if you already know that you will not need the job log and you want to conserve system resources. When you specify that you do not want to produce a job log, the job log will not be produced and remains in pending until removed either by the Remove Pending Job Log (QWTRMVJL) command or the End Job (ENDJOB) command.

To prevent the production of a job log, use the following instructions:

1. In IBM Navigator for i, open the **Job Properties - Job Log** window.  
(**Work Management > Active Jobs (or System Jobs) > Right-click the job > Properties > Job Log tab**)
2. Uncheck the **Produce a job log** box and click **OK**.

#### **Related concepts**

##### How job logs are created

The job logs are available when needed, but no work is done to produce job logs for which there is no need.

#### **Related tasks**

##### Deleting job log output files

Job logs are removed from the system when a job completes normally, or when the Remove Pending Job Log (QWTRMVJL) API or the End Job (ENDJOB) command is issued. Additionally if "clear incomplete job logs" is specified on the IPL, all of the jobs in job log pending are removed from the system during an IPL. Any remaining job log output files can be found under **Basic Operations > Printer Output**.

##### Controlling batch job log information



For your batch applications, you may want to change the amount of information logged. The log level (LOG(40 \*NOLIST)) specified in the job description for the IBM-supplied subsystem QBATCH supplies a complete log if the job abnormally ends. If the job completes normally, no job log is produced.

## Controlling information in a job log

When working with problems, you might want to record the maximum amount of information for jobs that have frequent problems. Alternatively, you might not want to create a job log for jobs that completed normally. Or you might want to exclude informational messages.

You can control what information is added to the job log by setting the message level, message severity or the message text level values in the job description. However, if you want to control what information is written to the job log of a specific job, use the **Job Properties - Job Log** window in IBM Navigator for i.

This window allows you to control the following:

- Whether the job log is produced and what method is used to produce it
- What to do when the maximum size is reached
- Whether to log commands from CL programs
- Whether to keep the messages in the job log and what specific messages should be kept (logging level and message severity)
- Whether to create printer output for a job log if the job ends normally and what to print

To access the Job Properties - Job Log window, follow the following steps:

1. From IBM Navigator for i, open the Job Properties window of the job and click the **Job Log** tab.  
**Work Management > Active Job > Right-click the job > Properties.**
2. For a detailed explanation of the different options that are available on this window, refer to the online help.

### Related concepts

[How job logs are created](#)

The job logs are available when needed, but no work is done to produce job logs for which there is no need.

### Related tasks

[Cleaning up job log pending](#)

There are a few ways to clean up, or remove jobs from job log pending. You can end the job with a value of 0 for the Maximum log entries (LOGLMT) parameter. If the job is already ended, you can run the Remove Pending Job Log (QWTRMVJL) API. You can also use the Work with Job Logs (WRKJOBLOG) command.

[Deleting job log output files](#)

Job logs are removed from the system when a job completes normally, or when the Remove Pending Job Log (QWTRMVJL) API or the End Job (ENDJOB) command is issued. Additionally if "clear incomplete job logs" is specified on the IPL, all of the jobs in job log pending are removed from the system during an IPL. Any remaining job log output files can be found under **Basic Operations > Printer Output**.

## Changing the log level of a job

The log level of a job is a numeric level assigned to a specific combination of message types that are logged. You can change the log level in the job description by using the character-based interface. However, if you want to change the log level of a specific job, use the **Job Properties - Job Log** window in IBM Navigator for i.

To access the **Job Properties - Job Log** window, follow the following steps:

1. From IBM Navigator for i, expand **Work Management > Active Jobs**.
2. Select a job and right-click **Properties**.
3. From the properties window of specific job, select the **Job Log** tab, and change the logging level.

### Related concepts

[Messages](#)

Messages contain the job name, the message type, the date and time it was sent, the action that occurred, and the necessary actions needed to fix a problem. This is useful when you are trying to troubleshoot any problems that might occur on your servers. You can access job logs for server jobs through IBM Navigator for i. Messages fall into two categories, alertable messages and messages logged in a job log.

#### Job logs

A job log contains information related to requests entered for a job. A job log has two forms, a pending form and a spooled form.

#### **Related tasks**

##### Cleaning up job log pending

There are a few ways to clean up, or remove jobs from job log pending. You can end the job with a value of 0 for the Maximum log entries (LOGLMT) parameter. If the job is already ended, you can run the Remove Pending Job Log (QWTRMVJL) API. You can also use the Work with Job Logs (WRKJOBLOG) command.

#### **Controlling batch job log information**

For your batch applications, you may want to change the amount of information logged. The log level (LOG(40 \*NOLIST)) specified in the job description for the IBM-supplied subsystem QBATCH supplies a complete log if the job abnormally ends. If the job completes normally, no job log is produced.

Controlling job logs at the job queue level (QBATCH) is done by adjusting the job log settings for the QBATCH subsystem job. You have the same options for controlling how job logs are produced at the subsystem job level as you do at the individual job level.

To adjust the job log settings for the job queue subsystem, do the following:

From IBM Navigator for i, open the **Subsystem Properties - Job Log** window for the job queue subsystem.

**(Work Management > Active Subsystems > QBATCH > Right-click the QBATCH job > Properties > Job Log tab)**

**Note:** If you uncheck the **Produce a job log field** field (\*PND) for the subsystem, the job log specific to the subsystem is not listed with the other printer output. You then need to use the Display Job Log (DSPJOBLOG) command to view the pending job log.

If the batch job is running a CL program, the CL program commands are logged only if the LOGCLPGM(\*YES) is specified on the Create Control Language Program (CRTCLPGM) command or the Change Program (CHGPGM) command.

#### **Related concepts**

##### Job logs

A job log contains information related to requests entered for a job. A job log has two forms, a pending form and a spooled form.

#### **Related tasks**

##### Deleting job log output files

Job logs are removed from the system when a job completes normally, or when the Remove Pending Job Log (QWTRMVJL) API or the End Job (ENDJOB) command is issued. Additionally if "clear incomplete job logs" is specified on the IPL, all of the jobs in job log pending are removed from the system during an IPL. Any remaining job log output files can be found under **Basic Operations > Printer Output**.

##### Stopping production of a specific job log

If you only want to stop the production of a particular job log, do not use the End Job Log Server (ENDLOGSVR) command. The ENDLOGSVR command ends all job log servers which results in stopping the production of all job logs.

##### Preventing the production of a job log

Preventing the production of a job log is useful if you already know that you will not need the job log and you want to conserve system resources. When you specify that you do not want to produce a job log, the job log will not be produced and remains in pending until removed either by the Remove Pending Job Log (QWTRMVJL) command or the End Job (ENDJOB) command.

## Deleting job log output files

Job logs are removed from the system when a job completes normally, or when the Remove Pending Job Log (QWTRMVJL) API or the End Job (ENDJOB) command is issued. Additionally if "clear incomplete job logs" is specified on the IPL, all of the jobs in job log pending are removed from the system during an IPL. Any remaining job log output files can be found under **Basic Operations > Printer Output**.

To delete job logs found in Printer Output, right-click the file name of the job log that you want to delete and click **Delete**.

### How to determine if it is safe to delete a job log

Balancing the decision of whether to keep job logs or delete them challenging. Job logs are things that you need to keep so you can troubleshoot a problem. Job logs are things that you do not want to keep because of they clutter up your system. When deciding which job logs to delete, or which job logs prevent from producing, consider the following guidelines:

- Is this a job that you can easily fix without looking at the job log?
- Is this a job that is similar to other jobs in the system? If it fails, are similar jobs likely to also fail? If so, then you might want to have only one of the jobs produce a job log.

### Related concepts

#### Job logs

A job log contains information related to requests entered for a job. A job log has two forms, a pending form and a spooled form.

### Related tasks

#### How to display job logs

You can see a job log from any place within work management that you access jobs, such as through the Subsystem area or the Memory Pool area. You can use IBM Navigator for i or the character-based interface to display job logs.

#### Ending the job log server

The End Job Log Server (ENDLOGSVR) command is used to end the job log server(s). The job log server writes job logs for jobs that are in a job log pending state. If more than one job log server job is active at the time this command is issued, all of the job log server jobs are ended.

#### Stopping production of a specific job log

If you only want to stop the production of a particular job log, do not use the End Job Log Server (ENDLOGSVR) command. The ENDLOGSVR command ends all job log servers which results in stopping the production of all job logs.

#### Preventing the production of a job log

Preventing the production of a job log is useful if you already know that you will not need the job log and you want to conserve system resources. When you specify that you do not want to produce a job log, the job log will not be produced and remains in pending until removed either by the Remove Pending Job Log (QWTRMVJL) command or the End Job (ENDJOB) command.

#### Controlling information in a job log

When working with problems, you might want to record the maximum amount of information for jobs that have frequent problems. Alternatively, you might not want to create a job log for jobs that completed normally. Or you might want to exclude informational messages.

#### Controlling batch job log information

For your batch applications, you may want to change the amount of information logged. The log level (LOG(40 \*NOLIST)) specified in the job description for the IBM-supplied subsystem QBATCH supplies a complete log if the job abnormally ends. If the job completes normally, no job log is produced.

## Producing printer output from job log pending

Jobs that do not have the IBM Navigator for i **Job Properties - Job Log** setting, **Produce a job log** field selected do not produce job logs. Instead the job log is in job log pending. To produce printer output from a job log that is in job log pending, use the character-based interface.

**Command:** Display Job Log (DSPJOBLOG)

### Related concepts

#### [Job log pending](#)

The job log pending state has been available for many years. When the job log attribute of a job is \*PND, no job log is produced. You can control how and under what circumstances the job log for a specific job is produced.

### Related tasks

#### [How to display job logs](#)

You can see a job log from any place within work management that you access jobs, such as through the Subsystem area or the Memory Pool area. You can use IBM Navigator for i or the character-based interface to display job logs.

## Cleaning up job log pending

There are a few ways to clean up, or remove jobs from job log pending. You can end the job with a value of 0 for the Maximum log entries (LOGLMT) parameter. If the job is already ended, you can run the Remove Pending Job Log (QWTRMVJL) API. You can also use the Work with Job Logs (WRKJOBLOG) command.

To end a job with **LOGMLT** set to 0, use IBM Navigator for i or the character-based interface.

### Related concepts

#### [Job log pending](#)

The job log pending state has been available for many years. When the job log attribute of a job is \*PND, no job log is produced. You can control how and under what circumstances the job log for a specific job is produced.

### Related tasks

#### [Controlling information in a job log](#)

When working with problems, you might want to record the maximum amount of information for jobs that have frequent problems. Alternatively, you might not want to create a job log for jobs that completed normally. Or you might want to exclude informational messages.

#### [Changing the log level of a job](#)

The log level of a job is a numeric level assigned to a specific combination of message types that are logged. You can change the log level in the job description by using the character-based interface. However, if you want to change the log level of a specific job, use the **Job Properties - Job Log** window in IBM Navigator for i.

### Related information

#### [Change Cleanup \(CHGCLNUP\) command](#)

#### [Exit Program for Tailoring Automatic Cleanup](#)

## *IBM Navigator for i*

1. Expand **Work Management** > **Active Jobs**.
2. Locate the job that you want to end.
3. Right-click the job and click **Delete/End**.
4. On the Confirm Delete/End window, set **Delete printer output** to No.
5. Complete the Confirm Delete/End window and click **Delete**.

## Character-based interface

**Command:**End Job (ENDJOB LOGLMT(0))

## Managing job accounting

The job accounting function is not active by default. It requires a few initial steps to set it up. The following information describes how to set up job accounting and perform some of the most common tasks associated with job accounting.

### Related concepts

#### Job accounting

The job accounting function gathers data so that you can determine who is using your system and what system resources they are using. It also assists you in evaluating the overall use of your system. Job accounting is optional. You must take specific steps to set up job accounting. You can request the system to gather job resource accounting data, printer file accounting data, or both. You can also assign accounting codes to user profiles or specific jobs.

### Related information

#### Journal Management

#### Set up journaling

## Setting up job accounting

To set up job accounting, use the character based interface.

1. Create a journal receiver. The journal receiver can be created with any name and library you choose. It is recommended to give it a name with a naming convention such as ACGJRN1 so that additional receivers (such as ACGJRN2, ACGJRN3) can be created with the Change Journal CHGJRN JRNRCV(\*GEN) command.

- a) **Command:** Create Journal Receiver (CRTJRNRCV)

```
CRTJRNRCV JRNRCV(USERLIB/ACGJRN1)
```

2. Create the job accounting journal. The journal name must be QSYS/QACGJRN, and you need authority to add objects to the QSYS library.

- a) **Command:** Create Journal (CRTJRN)

```
CRTJRN JRN(QSYS/QACGJRN) JRNRCV(USERLIB/ACGJRN1) AUT(*EXCLUDE)
```

The journal receiver should be the same as the receiver created in step 1. The authority can be set to anything you choose, but \*EXCLUDE is recommended since the data collected can be used to charge users for resource usage.

3. Change the journal accounting information (QACGLVL) system value. The system value can be set to journal job accounting information, or printer information, or both. \*JOB produces job (JB) journal entries, while \*PRINT produces direct print (DP) or spooled print (SP) journal entries. A value of \*NONE means no journaling is done for journal QACGJRN. Job accounting data will only be journaled for jobs that are started after the system value has been set to a value other than \*NONE.

- a) **Command:** Work with System Values (WRKSYSVAL) or Change System Value (CHGSYSVAL)

```
CHGSYSVAL SYSVAL(QACGLVL) VALUE('*JOB *PRINT')
```

4. Set the accounting code parameter ACGCDE for each user profile. The accounting code can be set to any alphanumeric string up to 15 characters in length. If determining the current user is important to your analysis of a job accounting journal entry, it is recommended that you set the ACGCDE parameter to the user profile's name.

- a) **Command:** Change User Profile (CHGUSRPRF) or Create User Profile (CRTUSRPRF)

```
CHGUSRPRF USRPRF(USERID1) ACGCDE(USERID1)
```

The accounting code can also be specified for a group of users by using the Change Job Description (CHGJOB) or Create Job Description (CRTJOB) commands.

The default accounting code for job descriptions is \*USRPRF, which means it uses the accounting code from the job's user profile. If a value other than \*USRPRF is specified in the job description, it will take precedence over the accounting code specified in the user profile

### Related concepts

#### About the accounting code

The initial accounting code (up to 15 characters in length) for a job is determined by the value of the ACGCDE (accounting code) parameter in the job description and user profile for the job.

## Controlling the assignment of accounting codes

An important aspect of any data processing application is ensuring that the correct control fields are specified. For job accounting codes, this can require a complex validity-checking function that not only checks for the existence of authentic codes, but also checks which users are allowed to use specific codes.

Accounting codes can be assigned in the following areas:

- User profile
- Job description
- In a job (Change Accounting Code (CHGACGCDE) command)

If it is important to control the assignment of accounting codes, consider the following:

1. Before an accounting code is placed in a user profile, ensure that the code is valid for a particular user.
2. Control the changing of accounting codes on the Change Job Description (CHGJOB) command by giving only the security officer authority to the CHGACGCDE command.
  - Or, use the CHGACGCDE command to allow users to change the job accounting code of their own or another job. To change another job, the user must also have the special authorization of \*JOBCTL.
3. Use a CL program and command to prevent changing accounting codes for a job on the job queue or for one job to change the accounting code of another job.  
For example, the CHGACGCDE command would be privately authorized and included in a CL program where it only changed the current job (such as when JOB(\*) is specified). The command would be authorized appropriately.

### Related concepts

#### Security and job accounting

Only the security officer (or a program adopting his authority) or a user with \*ALLOBJ and \*SECADM authority can change the Journal accounting information (QACGLVL) system value.

#### About the accounting code

The initial accounting code (up to 15 characters in length) for a job is determined by the value of the ACGCDE (accounting code) parameter in the job description and user profile for the job.

## Displaying the data collected

After collecting data in the job accounting journal, you can write the journal entries to a file and display them.

To do this, follow these steps:

**Note:** In the following example, the job accounting journal name is QACGJRN.

1. Create a copy of the system supplied model outfile for the accounting journal. QAJBACG4 is the model outfile for the \*TYPE4 outfile format.
  - a) **Command:** Create Duplicate Object (CRTDUPOBJ)

```
CRTDUPOBJ OBJ(QAJBACG4) FROMLIB(QSYS) OBJTYPE(*FILE) TOLIB(QTEMP)
NEWOBJ(MYJBACG4)
```

2. Dump the journal entries to the outfile that you just created. In the following example only the 'JB' or job type journal entries are being dumped.

a) **Command:** Display Journal (DSPJRN)

```
DSPJRN JRN(QACGJRN) ENTYP(JB) OUTPUT(*OUTFILE) OUTFILFMT(*TYPE4)
OUTFILE(QTEMP/MYJBACG4)
```

3. Start an SQL session. Then use the SELECT command from within the SQL session to choose the fields you want to display.

a) **Command:** Start Structured Query Language (STRSQL)

```
STRSQL
SELECT JAJOB, JAUSER, JAUSPF, JACDE, JACPU FROM QTEMP/MYJBACG4
```

You can display a list of field names interactively or to a file by creating and running a query using the Work with Queries (WRKQRY) command.

## Converting job accounting journal entries

You can use the OUTFILE parameter on the Display Journal (DSPJRN) command to write the job accounting journal entries into a database file that you can process.

The OUTFILE parameter allows you to name a file or member. If the member exists, it is cleared before the records are written. If the member does not exist, it is added. If the file does not exist, a file is created using the record format QJORDJE. This format defines the standard heading fields for each journal entry, but the job accounting data is defined as a single large field.

To avoid having to process the accounting data as a single large field, two field reference files are supplied to help you in processing the job accounting journal entries. The file QSYS/QAJBACG4 contains the record format QAWTJAJ4 and is used for JB entries. File QSYS/QAPTACG5 contains record format QSPJAPT5 and is used for DP or SP entries. The same format is used for all printer file entries regardless if the output is SP (spooled) or DP (nonspooled). The DP entry for directly printed files contains some fields that are not used; these fields contain blanks.

The following are some approaches you might use:

- The basic JB entries and DP or SP entries can be processed by creating two output files using the supplied field reference file formats and running the DSPJRN command once for JB and once for DP or SP. This allows you to define a logical file over the two physical files and use an high-level language program to process the externally described file.
- You can process only the JB entries by creating a file using one of the supplied field reference files (QSYS/QAJBACG4) to create an externally described file. This file can then be processed by the query utility or an high-level language program.
- You can convert both types of journal entries using the default DSPJRN format of QJORDJE. You can then use a program-described file to process the journal entries in a high-level language program.

The following DDS defines a physical file for the JB journal entries using the QAJBACG4 field reference file in QSYS. You can create the file (using the Create Physical File (CRTPF) command) with the same name (QAJBACG4) as the model file.

```
R QAWTJAJ4 FORMAT(QSYS/QAJBACG4)
```

The following DDS defines a physical file for the DP or SP journal entries using the QAPTACG5 field reference file in QSYS. You can create the file (using the CRTPF command) with the same name (QAPTACG5) as the model file.

```
R QSPJAPT5 FORMAT(QSYS/QAPTACG5)
```

You can specify a key field in either physical file; however, in this example, a logical file is used for sequencing. If you create two physical files (one for JB and one for DP or SP) with the members of the

same name, you can issue the following DSPJRN commands to convert the entries. Assume that you have created the physical files with the same names as the model files in your library YYYY.

```
DSPJRN JRN(QACGJRN) JRNCDE(A) ENTTP(JB)
OUTPUT(*OUTFILE) OUTFILE(YYYY/QAJBACG4)
DSPJRN JRN(QACGJRN) JRNCDE(A) ENTTP(SP DP)
OUTPUT(*OUTFILE) OUTFILE(YYYY/QAPTACG5)
```

You can control the use and selection criteria of the DSPJRN command so that you do not convert the same entries several times. For example, you can select all entries in a specific range of dates. You can convert all of the entries at a cutoff point for your job accounting analysis, for example, monthly. One or more journal receivers might have been used during the month. Note that each use of the DSPJRN command to the same member causes the member to be cleared before any new entries are added. Do not use the JOB parameter of the DSPJRN command as some entries are made for a job by a system job and will therefore not appear as you expect them to.

### Allowing the Processing of Both Physical Files:

Enter the following DDS to create a logical file to allow processing of both physical files. This allows you to read a single file in accounting code order and print a report using a high-level language program:

```
R QAWTJAJ4 PFILE(YYYY/QAJBACG4)
K JACDE
R QSPJAPT5 PFILE(YYYY/QAPTACG5)
K JACDE
```

### Processing Basic Job Accounting Record:

If you want to use a logical file to process only the basic job accounting record in accounting code order by a user name, you can enter the following DDS for a logical file:

```
R QAWTJAJ4 PFILE(YYYY/QAJBACG4)
K JACDE
K JAUSER
```

This logical file can be processed by the query utility or by a high-level language program. If an abnormal system ending occurs, the qualified job name in the first 30 bytes of the JARES field in the journal entry describe the system job that wrote the entry at the next IPL and not the job that used the resources. For this reason, any analysis done on the JB entries should use the JAJOB, JAUSER, and JANBR fields.

## Recovering and job accounting

If a job ends abnormally, the final accounting entry is written and all previously written accounting entries appear in the journal. If an abnormal system ending occurs, the following accounting data is lost to the last routing step or last end-of-accounting segment, whichever occurred most recently.

- Information on the number of lines and pages printed
- Number of files created
- Database put, get, and update operations
- Communications read and write operations
- Auxiliary I/O operations
- Transaction time
- Number of transaction fields
- Time active
- Time suspended

After an abnormal system ending, the job completion time in the journal is not the same as that in the CPF1164 message. The message uses the time nearest to that of the system ending, but the job accounting journal entries are sent to the journal during IPL, and the job completion time is the current system time, which is later than the time when the abnormal system ending occurred.



If the system ends abnormally, some journal entries can be lost. These are the entries that are written to the journal but not forced to disk (this is equal to FORCE(\*NO) on the Send Journal Entry (SNDJRNE) command). They include the following:

- JB entries caused by a Change Accounting Code (CHGACGCDE) command
- DP and SP entries

Whenever a job completes, the last accounting code entry is forced to disk (as if FORCE(\*YES) were specified on the SNDJRNE command). Whenever an accounting entry is forced to disk, all earlier entries in the journal, regardless of which job produced them, are forced to disk.

## Exception

If only \*PRINT accounting is specified on the system, there is not any job ending FORCE(\*YES) journal entries done. therefore, if a critical accounting entry is written by a CHGACGCDE command you want to ensure it is not lost in case of an abnormal system ending, you can issue a SNDJRNE command and specify the FORCE(\*YES) option. If files are also to be journaled to the accounting journal, any database changes are always forced to the journal, and this causes all earlier accounting entries to also be forced.

If an abnormal system ending occurs or you change an accounting code of a job other than your own, the qualified job name in the first 30 bytes of the JARES field in the journal entry describe the system job that wrote the JB entry at the next IPL and not the job that used the resources. The JAJOB, JAUSER, and JANBR fields should be used for analysis purposes.

## ***Damaged job accounting journal or journal receiver***

If damage occurs to the journal or to its current receiver so that the accounting entries cannot be journaled, a CPF1302 message is sent to the QSYSOPR message queue, and the accounting data is written to the QHST log in the CPF1303 message. The job trying to send the journal entry continues normally. Recovery from a damaged journal or journal receiver is the same as for other journals.

The journal QACGJRN should not be allocated by another job. If the journal is allocated by another job, the journal entry is changed to message text and sent to the QHST log as message CPF1303.

You can use the OUTFILE parameter on the Display Journal (DSPJRN) command to write the accounting journal entries to a database file that you can process.

You can also use the Receive Journal Entry (RCVJRNE) command on the QACGJRN journal to receive the entries as they are written to the QACGJRN journal. If the job accounting journal or journal receivers become damaged, the system continues to operate and to record accounting data in the history log. To recover from the journal or journal receiver damage, use the Work with Journal (WRKJRN) command. After recovering the damaged journal or journal receiver, change the Journal accounting information (QACGLVL) system value to a value appropriate for your installation. (Unless you change the QACGLVL system value, the system does not record accounting information in the new journal receiver.)

## ***Accessing the CPF1303 Message***

To access information from the CPF1303 message, create a high-level language program.

To define records that match the CPF1303 message, include the following fields:

**System Time** Char (8)  
**Message Record Number** Bin (4)  
**Qualified Job Name** Char (26)  
**Entry Type (JB, DP, or SP)** Char (2)  
**Length of Data** Bin (2)

Followed by fields:

JAJOB through JASPN for JB entries  
JAJOB through JABYTE for SP and DP entries

For an example program, refer to the section in the CL Programming book that discusses processing the QHST file for the job completion message.

The CPF1164 message always consists of three records and the CPF1303 message always consists of four records. The information contained in the standard journal prefix fields is not included in this message. All that is needed is information pertaining to the job end, date, and time. This information can be found in record 1 of the CPF1303 message.

## Managing workload groups

Workload groups provide the capability to manage work on a system.

The workload groups function can be used to limit the processing capacity of a workload to a subset of processor cores in a partition. A workload group can be created with a limit on the number of processor cores. Jobs can then be assigned to the workload group. The system enforces this processing core assignment by ensuring that any job and its associated threads cannot run on more processor cores than have been designated by the workload group.

Workload groups can be used to get better control of a workload and to ensure products only use a designated number of processor cores. Additionally, software vendors can employ workload groups to support sub-LPAR licensing. To take advantage of the enhanced licensing controls for products, IBM i License Management must be used to register and manage the enforcement of workload groups.

Collection Services, Performance Explorer, and Job Watcher have performance metrics that can help you manage and understand the performance of jobs running in a workload group.

### Example of workload groups use

A user has a multithreaded batch job that is CPU intensive. The user must run this job during the day but cannot afford to affect the performance of the production system. Assigning this batch job to a workload group puts the job into a 'processing container'. A workload group ensures that this job is kept to a limited amount of system processing capacity. If the workload group has a processor core limit of one, the batch job and any threads running under that job can only run on a single processor core. If this job is running on a multiple threaded core, multiple threads can be running for that designated batch job, but only a single core can be used at a time. This same concept applies to jobs running under a subsystem that has been assigned to a workload group. In this case, all jobs and their associated threads are limited to the number of processor cores specified in the group.

### Related information

[Collection Services](#)

[Planning for software licensing](#)

[JS \(Job Change\) journal entries](#)

[Retrieve Thread Attribute \(QWTRTVTA\) API](#)

[Change Job \(QWTCHGJB\) API](#)

## Setting up workload groups

A workload group defines the number of processor cores that can be used concurrently by jobs and threads that are associated with the group. Product entries can be added to a workload group to define the license term and feature of the product in the group. To set up workload groups, use the character-based interface.

1. Add a workload group using the [Add Workload Group \(ADDWLCGRP\) command](#). The workload group can be added with any name you choose. The processor limit defines the number of cores that jobs and threads associated with this workload group can run on concurrently.

```
ADDWLCGRP WLCGRP(MYGROUP) PRCLMT(2)
```

2. Add product information to the group using the [Add WLC Product Entry \(ADDWLCPRDE\) command](#). Product entries define the license term and feature of the product that is limited by the number of processor cores defined for the workload group. See [Setting up software licensing with workload groups](#) for more information on setting up software licensing with workload groups.

3. Add the workload group name to the subsystem description.

```
CHGSBSD SBSDB(MYLIB/MYSBSNAME) WLCGRP(MYGROUP)
```

4. Start the subsystem using the [Start Subsystem \(STRSBS\)](#) command. The workload group name can be changed while the subsystem is active, but has no effect on the jobs that are already active in that subsystem.

```
STRSBS SBSDB(MYLIB/MYSBSNAME)
```

**Note:** A CPI146C message (Subsystem &1 is using workload group &2) is logged in the subsystem job log when the subsystem is started. Jobs that start in subsystem MYSBSNAME are now limited to two processor cores, as defined by the workload group named “MYGROUP”.

5. To change the workload group for a job after it has already started, use the [Change Job \(CHGJOB\)](#) command. The job does not need to be running in a subsystem that has a workload group defined.

```
CHGJOB JOB(123456/QUSER/MYSERVER) WLCGRP(MYGROUP)
```

6. To change the processing capacity for a group, use the [Change Workload Group \(CHGWLCGRP\)](#) command. The processor limit can be changed while jobs using the workload group are active.

```
CHGWLCGRP WLCGRP(MYGROUP) PRCLMT(4)
```

### Notes:

- The [Remove Workload Group \(RMVWLCGRP\)](#) command can be used to change all jobs actively running in a workload group to no longer be assigned to that group.
- If you remove a workload group (RMVWLCGRP) while a subsystem that is using it is active, new jobs that are started are not limited.
- System jobs and subsystem jobs do not run in a workload group. They continue to use all processor cores available.
- The controlling subsystem (as defined by QCTLSBSD system value) cannot use a workload group.
- Batch immediate jobs are started using the workload group used by the job that starts them (parent job).
- Up to 255 workload groups can be created for the current system or logical partition.
- Changes to the number of processor cores for a workload group take effect immediately.

## Displaying workload groups

To display the workload groups defined on a partition, use the [Display Workload Group \(DSPWLCGRP\)](#) command.

1. Display the workload groups defined on a partition using the [Display Workload Group \(DSPWLCGRP\)](#) command.

```
DSPWLCGRP WLCGRP(*ALL) OUTPUT(*)
```

The current workload groups and any product entries associated with the groups are displayed. The output can also be sent to a spooled file by specifying OUTPUT(\*PRINT).

To display the workload group being used by a job, you can use the DSPJOB command (option 2, Display job definition attributes). You can also use the Retrieve Thread Attributes (QWTRTVTA) API.

## Auditing workload groups

A JS (Job Change) journal entry is written to the QAUDJRN journal when starting, ending, or changing a job. The workload group name is added to the JS audit entry at offset 3666 when the entry type is C, E,

or S. The field is 10 characters in length. This field continues to contain the Exit Job Name when the entry type is J, K, or L.

**Messages:**

- A CPI146C message is sent to the subsystem job log when the subsystem is started with a workload group defined.
- A CPI146D message is sent to QHST if an error occurs while trying to start a job in a subsystem that has a workload group defined.

## Reference

---

You might need to refer to these useful topics while using work management.

**(IBM i Information Center, Version 7 Release 2 (7.2) > Systems management > Work management > Reference)**

### **Server job table**

You can use this server table as a reference to find out how servers, server jobs, job descriptions, and subsystems are mapped to one another.

### **System value finder**

Use the system value finder to find information about system values. You can search for categories of system values as they appear in IBM Navigator for i or for the system value names you used in the character-based interface.

### **Work Management APIs**

The work management APIs perform functions that are used in a wide variety of applications. The Work Management APIs page displays a list of APIs that retrieve and manipulate jobs, subsystem storage pools, subsystem job queues, data areas, network attributes, system status, system values, and flight recorders. Also included are a list of Work Management exit programs.

### **IPL SRC finder**

Use the IPL system reference code (SRC) finder to find information about SRC messages that are displayed on your system when you perform an IPL. The SRCs indicate the status of the IPL and are often useful in problem analysis. You can search for an SRC by name, or display a list of the most common SRCs.

## Group jobs

The following information about group jobs is included as reference material for maintenance of older environments. In today's computing environment, it is typical for a single workstation to have separate sessions for separate functions.

The group jobs are similar to secondary interactive jobs requested by pressing the System Request key; however, up to 16 group jobs can be started for each sign on at a workstation (32 total when there is a secondary interactive job) and the application program can handle interruptions more easily.

### **Group job benefits**

The following lists some of the benefits of group jobs.

- The workstation user can press the Attention key to interrupt work in one interactive group job, change to any of several other interactive group jobs, and return to the original group job quickly. The Attention key is made valid by the Set Attention Program (SETATNPGM) command and can be used independently of group jobs.
- Using group jobs with display station pass-through provides a convenient and fast way to change among many interactive jobs on many different systems in a network.

### **Group job concepts**

- Group jobs apply only to interactive jobs.

- Up to 16 group jobs can exist in one group (16 more are available if the user transfers to a secondary interactive job).
- Group jobs are unique to a user (they are not shared by multiple users).
- Only one group job at a time is active (the others are suspended).
- Each group job is independent and has its own job log, spooled files, library QTEMP, and so forth.
- A group job is called by the Transfer to Group Job (TFRGRPJOB) command. This command is typically run from a user-written menu program, which is called by pressing the Attention key (the SETATNPGM command must have been previously run).
- A 512-byte group data area can be used to pass data between one group job and another. This group data area is implicitly created by the Change Group Attributes (CHGGRPA) command. The CL Programming book contains more information on group data areas.

## Changing to and from a group job

To change a nongroup job to a group job and to change a group job back to a nongroup job (if it is the only job in the group), use the Change Group Attributes (CHGGRPA) command.

## Creating a new group job

To create a new group job, use the Transfer Group Job (TFRGRPJOB) command.

**Note:** After each use of the TFRGRPJOB command, the SETATNPGM command must be used to set the Attention key on, if required.

## Transferring from one group job to another

To transfer from one group job to another group job in the same group, use the Transfer Group Job (TFRGRPJOB) command.

### Note:

1. After each use of the TFRGRPJOB command, the SETATNPGM command must be used to set the Attention key on, if required.
2. If you are in an update operation, use the Check Record Lock (CHKRCDLCK) command to check if the job has any record locks before transferring to another group job.

## Transferring control from one group job to another

You can transfer control from one group job to another if you have an Attention-key handling program. When the Attention key is pressed, an Attention-key-handling program can either present a menu (from which the user chooses a group job) or immediately transfer the user to another group job. Attention-key-handling support makes it easy to transfer control from one group job to another quickly, without ending one job to go to the other.

## Transferring to another group job without seeing a menu

You can use the Attention key to transfer directly to another job without seeing a menu. For example, the Attention-key-handling program for group job A could transfer to group job B. The Attention-key-handling program for group job B could transfer back to group job A. This allows a single keystroke to be used to switch between functions.

## Ending a group job

- To end one group job in a group, use the End Group Job (ENDGRPJOB) command.
- To end all group jobs in the group, use the SIGNOFF command.

**Note:** The ENDJOB command supports the parameter ADLINTJOBS. If \*GRPJOB is specified and the job specified on the JOB parameter is a group job, all jobs associated with the group end.

Additionally, the End Group Job (ENDGRPJOB) command does not support the signal SIGTERM. However, the End Job (ENDJOB) command does support the signal SIGTERM.

## Ensuring a normal group job end

In some environments it may be desirable to force the end user to correctly end certain group jobs rather than issuing the ENDGRPJOB command. For example, assume that the user may have a group job where there is a complex update involved and you want to be sure the job is ended normally. Another example is where the user may be in the middle of a SEU session and should complete the function normally.

It is possible to achieve this with the support given by the system. For example, you could use the following instructions:

1. Set a switch in the group data area that could be tested by each of the group jobs to function as the shutdown switch. That is, when the switch is set on, the group jobs function should be ended.
2. Access the active group job names by using the RTVGRPA command and the GRPJOB return variable.
3. Compare each name accessed (start with the second group job) against a predetermined list of the group job names that should be correctly ended.
4. If the group job name is not in the list, it can be ended immediately by the ENDGRPJOB command.
5. If the job must be correctly ended, transfer to the group job using the TFRGRPJOB command.

The Attention-key-handling program for all group jobs must be sensitive to the shutdown switch and would prevent transferring to another group job if the switch is set on.

If you have a controlling program for each of the group jobs that controls what happens when the user ends the function of the group job (for example, the update program), it could also test the shutdown switch and do a return. This ends the group job and returns control to the previous active group job.

The Attention-key-handling program can use the CHKRCDLCK command to determine if the workstation user pressed the Attention key when the application had a record locked for update. In this case, the attention program may send a message instructing the user to complete the operation before using the Attention key.

## Group job theory

The CHGGRPA command identifies the current job as a group job and gives it a group job name to uniquely identify it in the group. (At this point the group has only one group job.) Each group job is unique for a user. Two different users do not share the same group job. When a job is designated as a group job, it then has the capability to call a new group job. There are also restrictions on group jobs (such as RRTJOB, TFRJOB may not be used). When there is only one active job in the group, that job can become a nongroup job.

## Allowing group jobs to communicate

To allow group jobs to communicate with each other, a special 512-byte data area called a group data area is automatically created when a job becomes a group job. The group data area can only be accessed by jobs in the group by using the special value \*GDA in the DTAARA parameter of the data area command.

## Calling a group job

The use of group jobs does not require an Attention-key-menu approach as described in this section. A group job can be called from any application program or by the GRPJOB(\*SELECT) parameter on the TFRGRPJOB command.

## Group jobs and system request function

The Group Job function is similar to the System Request function in that there is only one job active at a time while the others are suspended. Group jobs differ from system request in the following ways:

- Starting a group job does not require signing on. The same user profile and environment are used.
- Up to 16 group jobs can exist at any one time. The user must select which group job to transfer to, whereas using system request permits the user to transfer between only two jobs. Normally in group jobs, a menu reached by pressing the Attention key allows the user to select which group job to transfer to. It is possible to use group jobs together with system request for a total of 32 group jobs available for a single user. However, these 32 jobs are in two separate groups, each group having its own group data area and other group attributes.
- The System Request function allows the workstation user to suspend a job while the keyboard is locked and application functions are in progress. This can interrupt a logical sequence of events. For example, records may be left locked. In contrast, the Attention key is active only when the keyboard is unlocked for input. Also, the application can control when the Attention key is active, and prevent its use at inappropriate times. The System Request function is always available if the workstation user has authority to it.

**Note:** The Presystem Request Program exit program is called when the user presses the System Request Key. The operating system calls the user-written exit program through the registration facility when the user presses the System Request key. One parameter is used for input and output. After the exit programs from the registration facility are called, the System Request menu is called based on the value that is returned in the System Request menu display flag. For additional information, see the System API Reference.

## Attention key handling program

You can identify a program as the Attention-key-handling program at a particular call level. The Attention-key-handling program runs in the same job and has the same job attributes, overrides, and group authorities as the program that issued the SETATNPGM command. However, program-adopted authority does not originate from the program that was interrupted. You may also specify an Attention-key-handling program in the user profile.

## Identifying a program as attention key handling

To identify a program as the Attention-key-handling program, use the Set Attention Program (SETATNPGM) command with SET(\*ON) specified. This command identifies this program at that call level in the job running the command. When the Attention key is pressed, the running job is interrupted, the display is saved, and the Attention-key-handling program is called. No parameters are passed to the Attention-key-handling program when it is called.

**Note:** The Pre-attention Program exit program is called when the user presses the System Attention key. The operating system calls the user-written exit program through the registration facility when the user presses the System Attention key. There are no input or output parameters. After the exit programs from the registration facility are called, the system attention program is called.

## Effect of call level on attention key status

The SETATNPGM command is call-oriented. That is, a SETATNPGM command issued at one call level causes the Attention-key-handling program to be in effect at the current call level as well as lower call levels, until another SETATNPGM command is run to change the Attention-key-handling program or Attention key status. Whenever a program that issued a SETATNPGM command returns, the display is restored and the Attention-key-handling program and Attention key status are reset to what they were before the current call. If a Transfer Control (TRFCTL) command is used instead of a RETURN command, the status is not reset until the program that was transferred to returns.

## When to use the attention key

Use the Attention key to call an Attention-key-handling program. In normal workstation use, the Attention key can be pressed only when the keyboard is unlocked; that is, the program is ready for input. This occurs when a read or write-read operation is issued or the UNLOCK DDS keyword is used in a write operation.

The use of the Attention key differs from that of the System Request key in that the application program has control over when it can be interrupted.

## Exception

An exception to this occurs with application programs performing a get-no-wait operation on multiple device files. Pressing the Attention key causes these programs to be interrupted at any point by the Attention-key-handling program. (Even though the input inhibited light may be on, the keyboard is unlocked during a get-no-wait operation.) Application programs performing sensitive functions (especially during a get-no-wait operation) should therefore be protected by running SETATNPGM PGM(\*CURRENT) SET(\*OFF) before and SETATNPGM PGM(\*CURRENT) SET(\*ON) after sensitive code.

**Note:** A high-level language program can use the SETATNPGM command by calling QCMDEXC.

## When not to use the attention key

The Attention key cannot be used to call an Attention-key-handling program when the following conditions exist:

- The keyboard is locked. (Note the exception described earlier for get-no-wait operations.)
- The System Request menu or any of its options is being used.
- The display message display is shown.
- The IBM i licensed program is already calling the Attention-key-handling program that makes it already active; however, if the program issues another SETATNPGM, the Attention key is enabled.
- A BASIC session is in progress, or a BASIC program is called.

## Attention key and BASIC session

In a BASIC session, the Attention key is handled by BASIC, as appropriate. For example, if a BASIC program is called after a SETATNPGM command has set the Attention key on, the Attention key is handled by BASIC. After the BASIC program ends, your Attention-key-handling program takes effect again.

## Attention key handling program coding tips

Caution is necessary when defining an Attention-key-handling program because the Attention-key-handling program runs in the same job as the program that is in progress when the Attention key is pressed. Therefore, the interrupted program is not protected by any locks it holds. If the interrupted program has an exclusive lock on an object, the Attention key program, because it runs in the same job, is part of the job that has the exclusive lock.

The following guidelines are recommended for defining Attention-key-handling programs:

- Use simple functions such as menus that allow the workstation user to transfer to another group job or to a secondary interactive job.
- Avoid referring to objects or functions that may be in use when the Attention key is pressed.
- Avoid calling non-recursive functions when the Attention key is pressed. Non-recursive functions are functions that cannot be interrupted and then called again. Many functions, such as high-level language programs and utilities like DFU, are non-recursive.
- Avoid giving an option that allows the workstation user to display the command entry display as part of the current job. For users who are programmers, it is meaningful to display a menu that includes an option for the command entry display. The command entry display should be specified as a separate



group job (for example, by specifying INLGRPPGM(QCMD) on the TFRGRPJOB command). This avoids re-using objects already in use.

- Attention-key-handling programs do not have the authority adopted by the program that was in progress before the Attention key was pressed.
- Attention-key-handling programs do not have their own data area (\*LDA). Since there is only one local data area per job, and the Attention-key-handling program runs in the same job as the interrupted program, both programs share the same local data area.
- Be aware that a read-from-invited devices operation could time-out during the time that the Attention-key-handling program is running. Therefore, if a time-out were to complete in the program in progress while the Attention-key-handling program is running, whatever action taken as a result of that time-out occurs on return to the program in progress. For example, if the following conditions are met, the program exits on return from the Attention key handler:
  - The WAITRCD value in the file is set to 60 seconds.
  - The program is set to exit if a key is not pressed in one minute.
  - The Attention key program is called and runs longer than that minute.

However, caution should be used, since a check for available data is made before checking that the time-out has completed. If a key is pressed immediately after leaving the Attention key handler, data could be available that could complete the read-from-invited devices and the time-out would not be checked. This could cause unexpected results.

## Group job performance tips

This topic provides you with some tips on maintaining good system performance when using group jobs.

- The effect on the system for a large number of suspended jobs is normally small if the dedicated main storage requirement is not a factor.
- When a TFRGRPJOB command runs and a new job must be started, the overhead involved is roughly the same as signing on to the system. When the command is run and the group job is already started, the overhead required is roughly the same as using the transfer to a secondary job option on the System Request menu when the secondary job is already active.
- If a group job is to be run with any frequency, it is desirable to prevent it from ending. That is, do not end the program, but issue a TFRGRPJOB command to prevent job starting each time the group job function is needed.
- The SETATNPGM command causes the current display to be saved when the Attention key is pressed, and to be restored when the Attention key-handling-program ends. This is roughly the same as using of the System Request menu and has a more noticeable effect on remote workstations.
- The controls on the number of jobs active in the system (the MAXJOBS parameter on the CRTSBSD command) are not affected by the number of group jobs active at any time.
- All system values that control the creation of job structures (QACTJOB and QADLACTJ, and QTOTJOB and QADLTOTJ) are affected; these values may need to be increased to allow for the addition of group jobs.

## Troubleshooting for work management

This topic helps you troubleshoot some of the most common problems that occur in work management.

### My job is hung

These tables list the possible reasons why a job might be hung.

<b>Job is waiting to get a lock on an object</b>		
	How to diagnose:	View the status of the job in IBM Navigator for i; see <a href="#">Determining the status of a job</a> . A job that is waiting to get a lock will have a status of <i>Waiting for lock</i> .
	Recovery:	View the list of locked objects for the job to determine which object the job is waiting to get a lock on. 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.

<b>Job is held</b>		
	How to diagnose:	View the status of the job in IBM Navigator for i; see <a href="#">Determining the status of a job</a> .
	Recovery:	Right-click the job and click <b>Release</b> .

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

<b>Job queue is held</b>		
	How to diagnose:	View the status of the job queue in IBM Navigator for i;
	Recovery:	<ol style="list-style-type: none"><li>1. Move the job to a job queue that is not held, see <a href="#">Moving a job to a different job queue</a>.</li><li>2. Release the job queue. To do so, right-click the job and click <b>Release</b>.</li></ol>

<b>Job queue has not been allocated by an active subsystem</b>		
	How to diagnose:	View the status of the job queue in IBM Navigator for i.
	Recovery:	<ol style="list-style-type: none"><li>1. Move the job to a job queue that is allocated by an active subsystem, see <a href="#">Moving a job to a different job queue</a>.</li><li>2. Start a subsystem which contains a job queue entry for this job queue, see <a href="#">Starting a subsystem</a>.</li><li>3. Add a job queue entry for this job queue to an active subsystem using the Add Job Queue Entry (ADDJOBQE) command.</li></ol>

<b>Subsystem maximum has been reached</b>		
	How to diagnose:	View the maximum active jobs value for the subsystem in IBM Navigator for i. To do so, right-click the subsystem and click <b>Properties</b> .
	Recovery:	<ol style="list-style-type: none"><li>1. Move the job to a different job queue, see <a href="#">Moving a job to a different job queue</a>.</li><li>2. Increase the maximum value. To do so, use the Change Subsystem Description (CHGSBSD) command.</li></ol>

<b>Job queue maximum has been reached</b>		
	How to diagnose:	View the maximum active jobs value for the job queue in IBM Navigator for i. To do so, right-click the job queue and click <b>Properties</b> . Then select the <b>Activity</b> tab.
	Recovery:	<ol style="list-style-type: none"> <li>1. Move the job to a different job queue; see <a href="#">Moving a job to a different job queue</a>.</li> <li>2. Increase the maximum value. To do so, use the Change Job Queue Entry (CHGJOBQE) command.</li> </ol>

<b>Maximum value for the priority level has been reached</b>		
	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 IBM Navigator for i. To do so, right-click the job queue and click <b>Properties</b> . Then select the Activity tab and click the <b>Advanced</b> button.
	Recovery:	<ol style="list-style-type: none"> <li>1. Move the job to a different job queue; see <a href="#">Moving a job to a different job queue</a>.</li> <li>2. Change the job queue priority of the job; see <a href="#">Specifying the priority for the job queue</a>.</li> <li>3. Increase the maximum value. To do so, use the Change Job Queue Entry (CHGJOBQE) command.</li> </ol>

## My job is experiencing poor performance

These are the possible reasons why a job might experience poor performance.

<b>Insufficient memory</b>		
	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 IBM Navigator for i, see <a href="#">Checking memory pool usage</a> . 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:	<ol style="list-style-type: none"> <li>1. Turn on the system tuner if you are not already using it. See <a href="#">Performance system values: Automatically adjust memory pools and activity levels for the information about automatically adjusting memory pools and activity levels</a>.</li> <li>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 might also want to check the machine pool to verify that the amount of memory being used is not affecting all jobs on the system.</li> </ol>

<b>Activity level too low</b>		
	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 <i>Waiting for activity level</i> , then view the properties of the memory pool in IBM Navigator for i, see <a href="#">Checking memory pool usage</a> . 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:	<ol style="list-style-type: none"> <li>1. Turn on the system tuner if you are not already using it. See <a href="#">Performance system values: Automatically adjust memory pools and activity levels</a> for the information about automatically adjusting memory pools and activity levels.</li> <li>2. Manually tune the pool by increasing the activity level for the memory pool.</li> </ol>

<b>Insufficient CPU resource</b>		
	How to diagnose:	View the CPU % column for the job and other jobs in the Active Jobs list of IBM Navigator for i. If the system is very busy, your job might not be getting enough CPU resource to complete its work.
	Recovery:	<ol style="list-style-type: none"> <li>1. If possible, end or hold unnecessary work on the system.</li> <li>2. If a few jobs are CPU intensive, change the run priority of these jobs (a higher run priority value equals a lower run priority for the job).</li> </ol>

<b>Memory pool paging option</b>		
	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 might be beneficial.
	Recovery:	The expert cache can be turned on in IBM Navigator for i by changing the Paging option for a shared memory pool to Calculated. The Paging option is located on the <b>Configuration</b> tab of the memory pool's <b>Properties</b> page and is only available on shared pools(not private pools).

<b>Low job run priority</b>		
	How to diagnose:	To determine the run priority of a job relative to other jobs on the system, see <a href="#">Viewing job attributes</a> .
	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 <a href="#">Viewing job attributes</a> . Also, on a system with high CPU utilization and a job with a low run priority, see <a href="#">Performance system values: Dynamically adjust job priorities within priority bands</a> and <a href="#">Performance system values: Dynamically adjust job priorities of interactive jobs</a> . The system values might be useful.

For more information about performance, see [Performance](#). If you want more information about how to tune performance on your system, see [Tuning performance](#).

## Prestart job investigation

This topic provides steps to help you answer the question, "How do I find the real user of a prestart job and determine the resources used by that prestart job?"

### IBM Navigator for i

You can use IBM Navigator for i work management views and monitors for a real time analysis of what is happening on your system.

1. Use the Server Jobs view to see the active server jobs and the current user. (**Work Management > Server Jobs**)
  - Open the Server Jobs list and select **Actions > Columns** and make sure that the Current User, Total CPU Time, and Total CPU DB Time are in the **Columns to be displayed** list.
  - If your active server job list is large, you can limit what is displayed by a job name, job number, current user, or status. Click **Actions > Include**.
  - You can sort the display order of the active server job list by clicking on the column headings.

Once you have located a job of interest, you can right-click the job and access the job's call stack, job log, elapsed performance statistics, last SQL statement, and the job's properties.
2. Set up a system monitor that monitors the overall CPU Utilization. (Expand **Monitors**, right-click **System**, and select **New Monitor**.)
  - While the monitor is running, you can click one of the points to view the next level of detail. For example, when monitoring CPU Utilization you can display a list of jobs that have the highest CPU utilization. You can then right-click a job that is using a lot of CPU and click **Properties** to display the job's properties. (See the online help for more information about how to use the system monitor.)

### Character-based interface

**Command:** Work with Active Job (WRKACTJOB)

This command displays the current user of the initial thread (which is the job when the job is single threaded). This is the same data that is shown in the GUI.

#### Related concepts

[Prestart job entries](#)

You define the prestart job by using a prestart job entry. A prestart job entry does not affect the device allocation or program start request assignment.

[Prestart jobs for servers](#)

In the prestart job model there is one primary listening job, generally called the daemon job or listener job, and multiple server jobs that process the client requests. The daemon job listens on the port for connection requests. When a new connection is received, the daemon does some general work, then gives the socket descriptor to a waiting prestart server job.

#### Related tasks

[Adding prestart job entries](#)

Prestart job entries identify prestart jobs that may be started when the subsystem is started or when the Start Prestart Jobs (STRPJ) command is entered. You can add prestart job entries to the subsystem description by using the character-based interface.

[Changing prestart entries](#)

You can change a prestart job entry in the specified subsystem description. The subsystem may be active when the prestart job entry is changed. Changes made to the entry when the subsystem is active are reflected over time. Any new prestart jobs started after the command is issued use the new job-related values. This command identifies prestart jobs that are started when the subsystem is started or when the Start Prestart Jobs (STRPJ) command is issued.

[Removing prestart job entries](#)

You can remove prestart job entries from the subsystem description by using the character-based interface. A prestart job entry cannot be removed if any currently active jobs were started using the entry.

## Related information for Work management

---

Other information center topic collections contain information that relates to the Work Management topic collection.

### **Experience reports**

The work management experience reports give you practical, real world ways to use the work management tools in your everyday tasks.

### **Networking**

Your understanding of networking technologies is a vital part of your company's total e-business solution. Learn how to connect your business to the Internet, configure e-mail, and serve multimedia objects to Web browser clients. You can integrate file and print services, user profile management, and network operations. Find information about the Windows server that can be integrated into the server, and read about security offerings that can help protect your resources.

### **Retrieve Network Attributes (QWCRNETA) API**

The Retrieve Network Attributes (QWCRNETA) API lets you retrieve network attributes.

### **Retrieve IPL Attributes (QWCRIPLA) API**

The Retrieve IPL Attributes (QWCRIPLA) API returns the settings of attributes that are used during the IPL. This API provides support similar to the Display IPL Attributes (DSPIPLA) command.

### **Performance**

Understanding all the different processes that affect system performance can be challenging for the inexperienced user. Resolving performance problems requires the effective use of a large suite of tools, each with its own unique set of requirements and supported functions. Even after you have gathered and analyzed performance data, knowing what to do with that information can be daunting. This topic will guide you through the tasks and tools associated with performance management.

### **Performance explorer**

Performance explorer collects more detailed information about a specific application, program or system resource, and provides detailed insight into a specific performance problem. This includes the capability both to perform several types and levels of traces and to run detailed reports.

### **Time management**

Within the time management component of IBM Navigator for i, you can work with the time zone and time adjustment functions. With these functions, you can choose a time zone for your system to use and adjust the system time.

### **System values**

System values are pieces of information that affect the system operating environment. System values are not objects on the system. Rather, system values contain control information for the operation of certain parts of the system.

## Notices

---

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

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

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing  
IBM Corporation  
North Castle Drive  
Armonk, NY 10504-1785  
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing  
Legal and Intellectual Property Law  
IBM Japan Ltd.  
1623-14, Shimotsuruma, Yamato-shi  
Kanagawa 242-8502 Japan

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

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

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

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

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation  
Software Interoperability Coordinator, Department YBWA  
3605 Highway 52 N  
Rochester, MN 55901  
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

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

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

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

#### COPYRIGHT LICENSE:

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

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs.

© Copyright IBM Corp. \_enter the year or years\_.

## Programming interface information

---

This Work management publication documents intended Programming Interfaces that allow the customer to write programs to obtain the services of IBM i.

## Trademarks

---

IBM, the IBM logo, and [ibm.com](http://ibm.com) are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "[Copyright and trademark information](http://www.ibm.com/legal/copytrade.shtml)" at [www.ibm.com/legal/copytrade.shtml](http://www.ibm.com/legal/copytrade.shtml).

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.



Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java and all Java-based trademarks and logos are trademarks of Oracle, Inc. in the United States, other countries, or both.

Other product and service names might be trademarks of IBM or other companies.

## Terms and conditions

---

Permissions for the use of these publications is granted subject to the following terms and conditions.

**Personal Use:** You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative works of these publications, or any portion thereof, without the express consent of IBM.

**Commercial Use:** You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.







Product Number: 5770-SS1