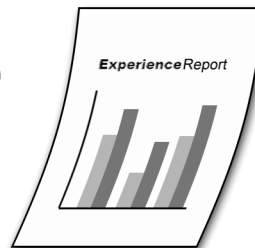


iSeries



# Subsystem configuration

# Experience Report





iSeries



# Subsystem configuration



---

# Contents

<b>Subsystem Configuration</b>	1
Why consider multiple subsystems?	1
Server subsystems	2
Multiple subsystem considerations	2
Server subsystem configuration	3
Server subsystem defaults	10
User-defined server subsystems	13
Managing server jobs	16
All server jobs	17
Server jobs by server	18
Server jobs by user	20
Ended server jobs	22
Server job table	25
Server job details and properties	25
<b>Interactive subsystem configuration</b>	39
Scenario details: CL program example	44
Managing interactive users and devices	53
Getting users to a specific subsystem	53
Assign device names	53
Manage inactive interactive sessions	55
Manage Device Recovery	56
Determining the IP Address for a Device Description	56
<b>References and resources</b>	57
<b>Disclaimer</b>	59



---

# Subsystem Configuration

The default subsystem configuration shipped with OS/400<sup>(R)</sup> is a basic subsystem configuration that works well for small systems. However, as the number of users increases on the system, it is desirable to split the work into multiple subsystems to better manage the work on the system.

This experience report includes the following information:

**“Why consider multiple subsystems?”**

This information discusses the various reasons you may want to perform subsystem configuration and add additional subsystems for your users.

**“Server subsystems” on page 2**

This information describes how to perform subsystem configuration for server jobs.

**“Managing server jobs” on page 16**

This information describes how to manage server jobs and see who is using a server job.

**“Interactive subsystem configuration” on page 39**

**This information describes how to perform subsystem configuration for interactive users.**

**“Managing interactive users and devices” on page 53**

**This information describes how to manage interactive user sessions and devices.**

---

## Why consider multiple subsystems?

OS/400<sup>(R)</sup> is shipped with a standard set of subsystems. The following subsystems are necessary for supporting user work:

- QBASE or QCTL, QINTER, and QCMN
- QSYSWRK
- QSERVER
- QUSRWRK

Interactive users typically run in QBASE or QINTER; which one you use is determined by the controlling subsystem system value (QCTLSBSD). This experience report assumes you are using QINTER as your interactive subsystem.

Server jobs run in QSYSWRK, QUSRWRK, and QSERVER. 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.

This experience report focuses on the users that run in QINTER and QUSRWRK. 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, including the following:

- Improves manageability of the work on the system due to better intellectual control over what work is running in what subsystems. For example, for server jobs, you may 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, etc. This allows you to understand of the type of work being done in the various subsystems.

- Provides the ability to disallow sets of users to access the system at certain periods of time. 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.
- Improves 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.
- Improves error tolerance by spreading the work across multiple subsystems; this is particularly important for interactive subsystems. For example, if a network failure occurs and many sessions enter into device recovery, the interactive subsystem can become very busy with the device recovery processing. This is because the subsystem job is the central job for managing device recovery. If hundreds of devices are affected by a network failure, they are recovered by the subsystem one at a time; such processing can negatively impact the ability for users to sign on or sign off the system. By splitting the users into multiple interactive subsystems, you can have more than one subsystem job manage the device recovery processing. Limiting the number of devices to 250 to 300 in one subsystem provides significantly improved availability and scalability of the interactive subsystem.
- Improves interactive subsystem startup time. When an interactive subsystem is started, it attempts to allocate all the devices defined by the workstation entries added to the subsystem description. As the number of devices increases, subsystem startup time may increase. You can keep subsystem startup times shorter by subdividing the work across multiple subsystems.
- Provides additional options for performance tuning. Several performance attributes, such as run priority, time slice, default wait, maximum CPU and others are performance attributes specified in the class description. Storage pools and maximum jobs are specified on the subsystem description. You can tune performance by using multiple routing entries with appropriate classes within a single subsystem, or you can simply set up multiple subsystems with a small number of routing entries. Either way, you give more system resources to users doing business critical work while fewer system resources are available for users doing work that is less important. Which approach you use depends upon your requirements. The detailed information later in this experience report uses multiple subsystems with simpler configurations.

---

## Server subsystems

The default subsystem configuration shipped with OS/400<sup>(R)</sup> 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.

This experience report includes the following information:

**“Multiple subsystem considerations”**

This section describes examples where using multiple subsystems is a good idea and describes some things to be aware of when creating and configuring subsystems.

**“Server subsystem configuration” on page 3**

This section describes how to perform subsystem configuration for server jobs.

**“Server subsystem defaults” on page 10**

This section describes how to modify server subsystem default values.

**“User-defined server subsystems” on page 13**

This section describes how to create a user-defined server subsystem.

## Multiple subsystem considerations

Server jobs are jobs that run continuously in the background on the iSeries<sup>(TM)</sup> server 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. Server jobs are most commonly associated with such functions as HTTP, Lotus Notes<sup>(R)</sup>, and TCP/IP. The Work Management: Server Jobs topic in the iSeries Information Center contains more information about server jobs.

There are several reasons why configuring the server subsystems for server jobs is advantageous. Some of those reasons are as follows:

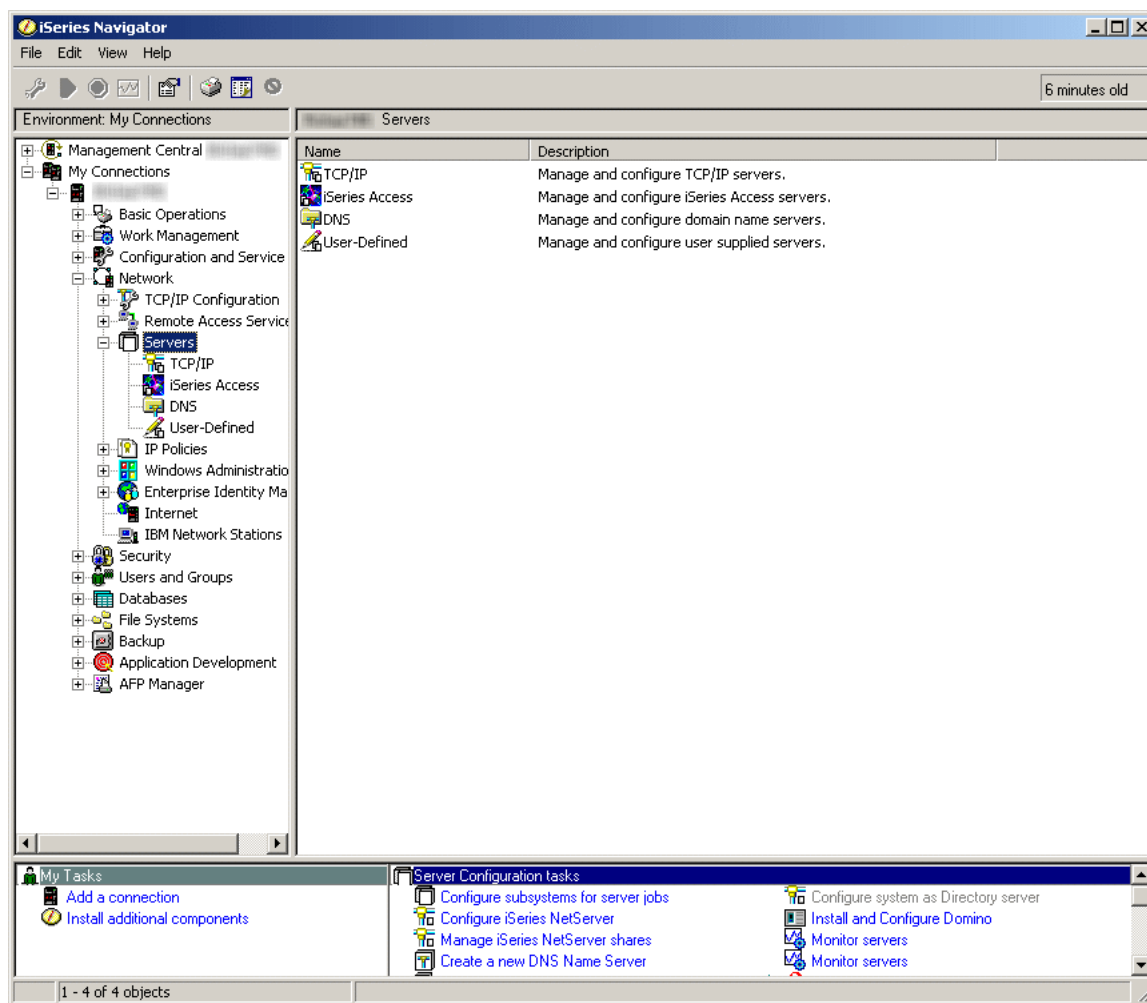
- Performance tuning is one reason to do subsystem configuration and can be done by changing routing entry information. The routing information that can be changed includes the memory pool that server jobs run in and class information. The class information is used to configure various performance attributes such as run priority, time slice, maximum threads allowed, default wait time, and more. Display Subsystem Description (DSPSBSD) command displays a menu that will give access to routing entry information. Additional performance tuning information can be found in the iSeries Information Center.
- System management is another reason to do subsystem configuration. Configuring servers to use multiple subsystems distributes system work and allows for greater control over system resources. For example, you may want to have a subsystem dedicated to database server work and let it have a large amount of system resources. Another example could be to control access to a subsystem by only allowing users on a particular subnet to send work to the subsystem. System management can also be used to aid in system maintenance. As a further example, you could have a subsystem for all employees on the east coast, a subsystem for employees in the Midwest, and another subsystem for the west coast employees. As the day progresses when the east coast employees are done working you can end the east coast subsystem without affecting the west coast and Midwest employees. By slowly taking down each subsystem as the respective employees leave, you can perform maintenance on those resources no longer in use without a large impact to users. For more information see the manage jobs and threads or the manage subsystems information found in the Work Management topic in the iSeries Information Center.
- Spreading the work across multiple subsystems provides improved error tolerance. By distributing work across multiple subsystems, the impact of a failure in one subsystem is more contained without affecting as many potential users. A subsystem is handling over 5000 jobs at a time, if something bad happens to that subsystem where it gets into an error state; 5000 jobs are affected and could be lost. If those 5000 jobs were spread out over multiple subsystems, the result of a failure in one of these many subsystems impacts a much smaller number of jobs.

## Server subsystem configuration

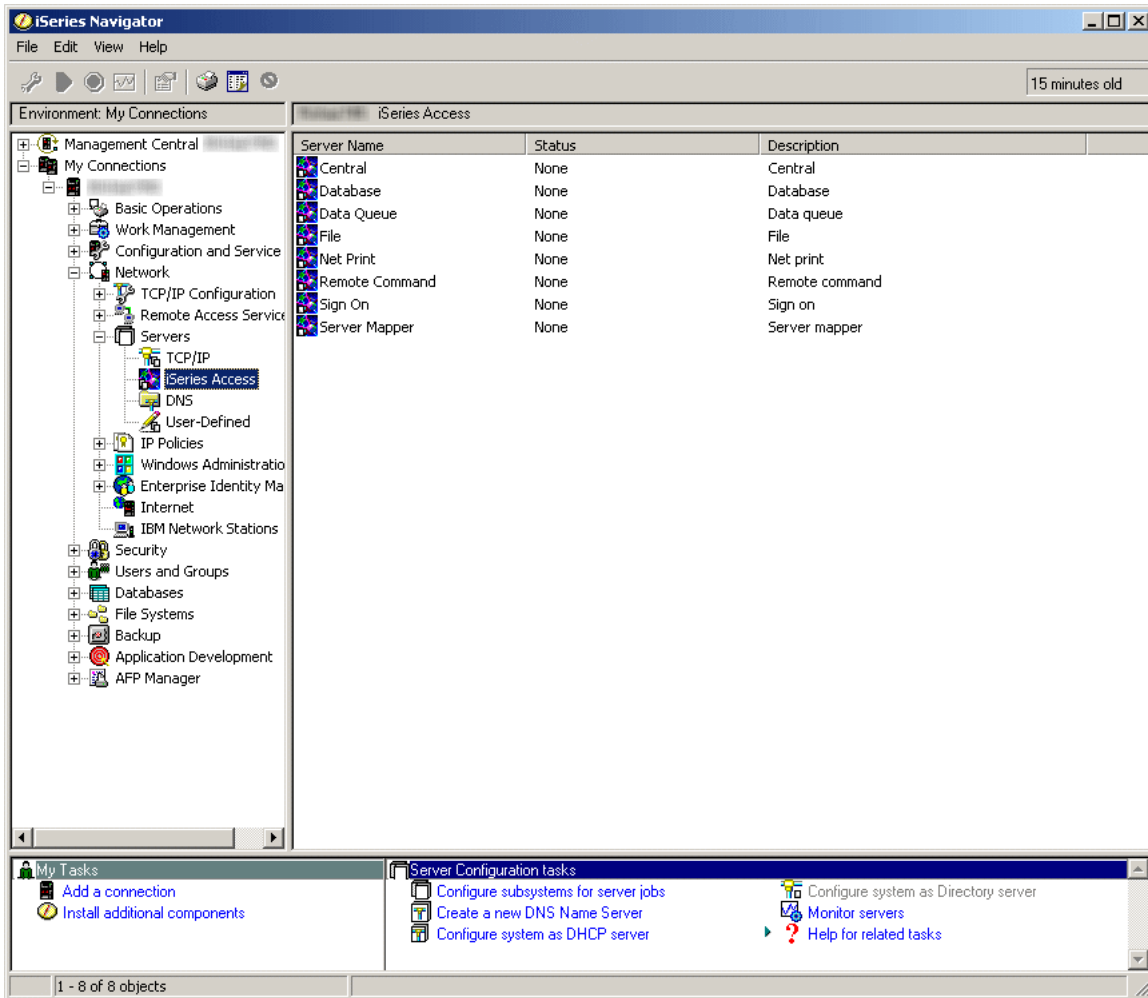
Some of the server jobs provided by OS/400<sup>(R)</sup> can be configured to run in subsystems other than the default subsystem. A server can use subsystems that are shipped with the system or “User-defined server subsystems” on page 13. A server can also use multiple subsystems. Routing to the proper subsystem is controlled by IP addresses.

To specify which subsystem server jobs use, you must use iSeries<sup>(TM)</sup> Navigator. The following steps show how to configure server subsystems using the iSeries Navigator.

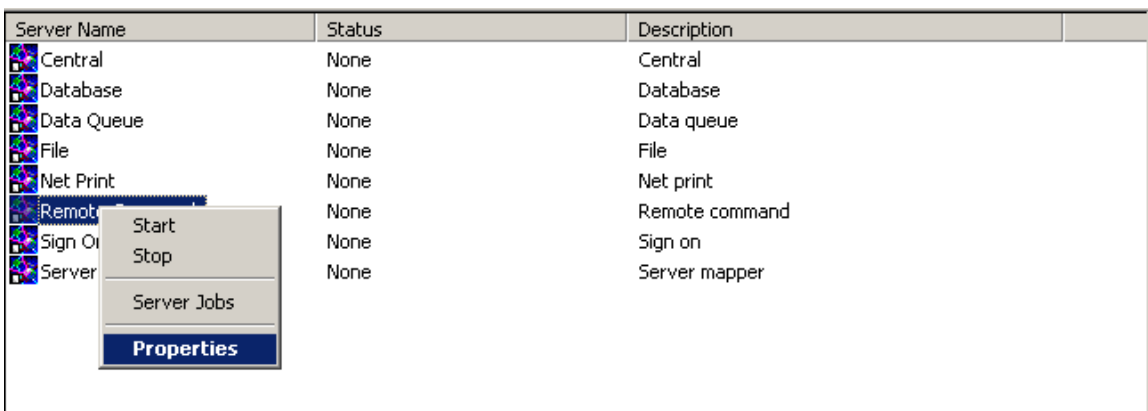
1. Open **iSeries Navigator**.
2. Expand **My Connections**.
3. Expand your **iSeries Server**.
4. Expand **Network**.
5. Expand **Servers**.



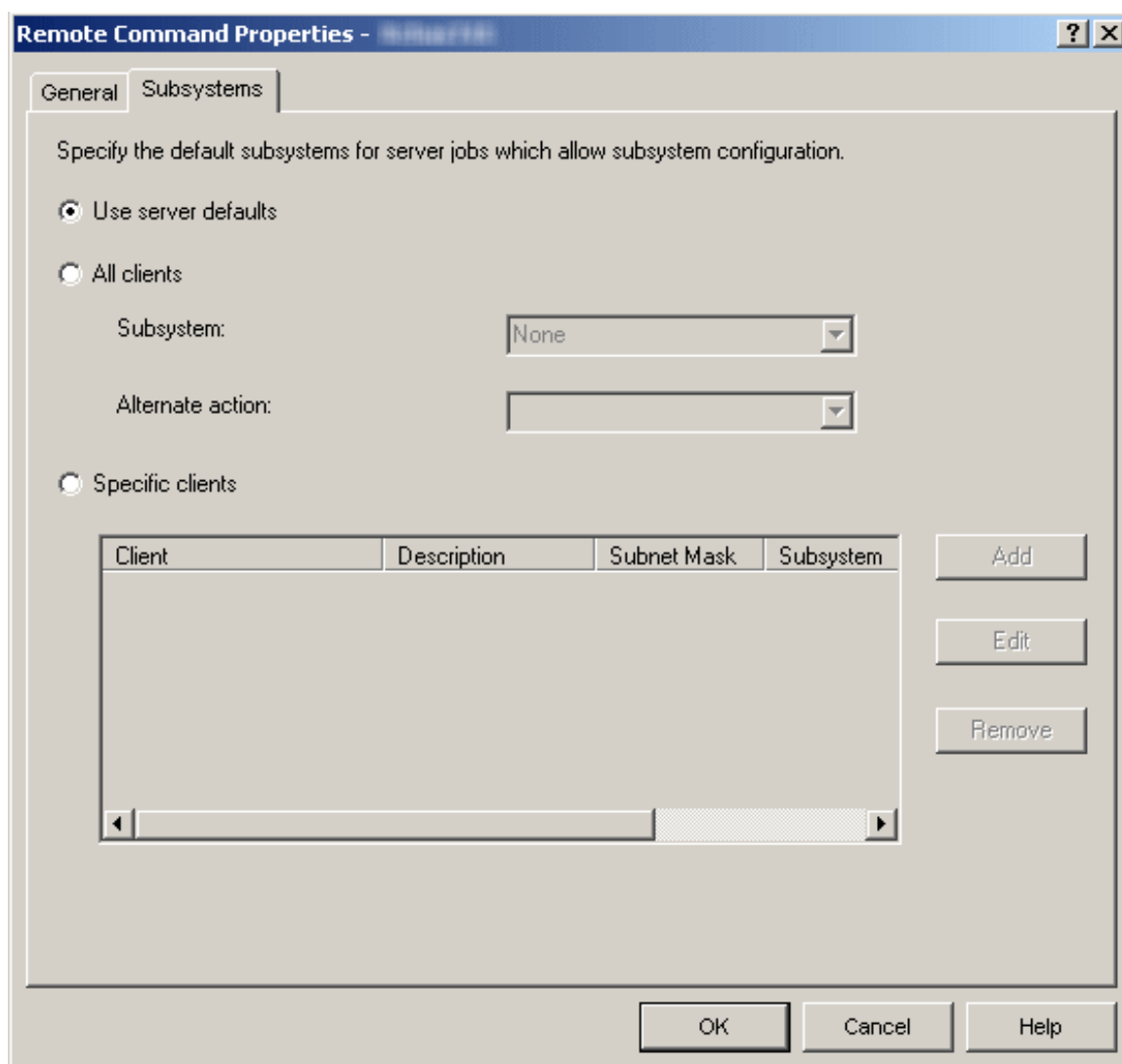
- Select the type of servers that you want to configure subsystems for. DDM and iSeries NetServer are **TCP/IP** servers. Central, Database, Data Queue, Files, Net Print, Remote Command, and Sign On are **iSeries Access** servers. The remaining steps use a subsystem from iSeries Access. The subsystem configuration steps are identical for each server.



- In the right pane, right-click on the server that you want to configure subsystems for and select **Properties**. Another window opens, showing tabs for general and subsystem information for that server.



- Select the **Subsystems** tab.



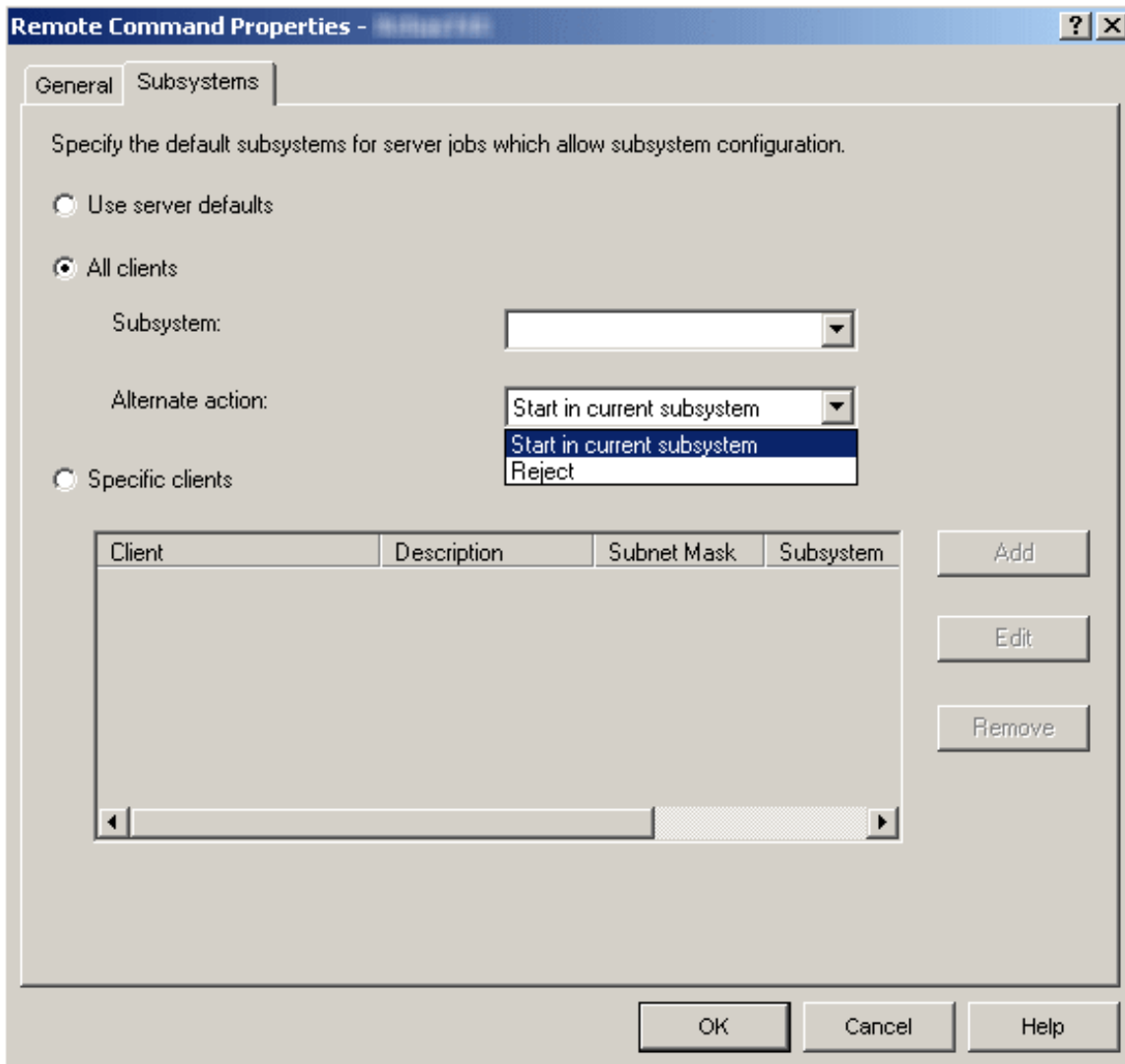
From the Subsystems tab you can specify in which subsystem you want this server's job to run. There are different ways to configure the subsystems. Select the appropriate configuration method and complete the following instructions for your selection:

- **Use server defaults**

Select this option if you want the particular server that you are working on to use the server defaults. 'Server defaults' allow you to configure in one place the general characteristics you want for subsystems for all server jobs. You can then have individual servers use these defaults, or you can have a more specific configuration for a specific type of server. See "Server subsystem defaults" on page 10 for more information on how to use and set the server defaults.

- **All clients**

Select this option if all clients that use this server are to use the same subsystem and alternate action. Using 'All clients' provides an easy way to subdivide the work done by the various servers into a subsystem per 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.

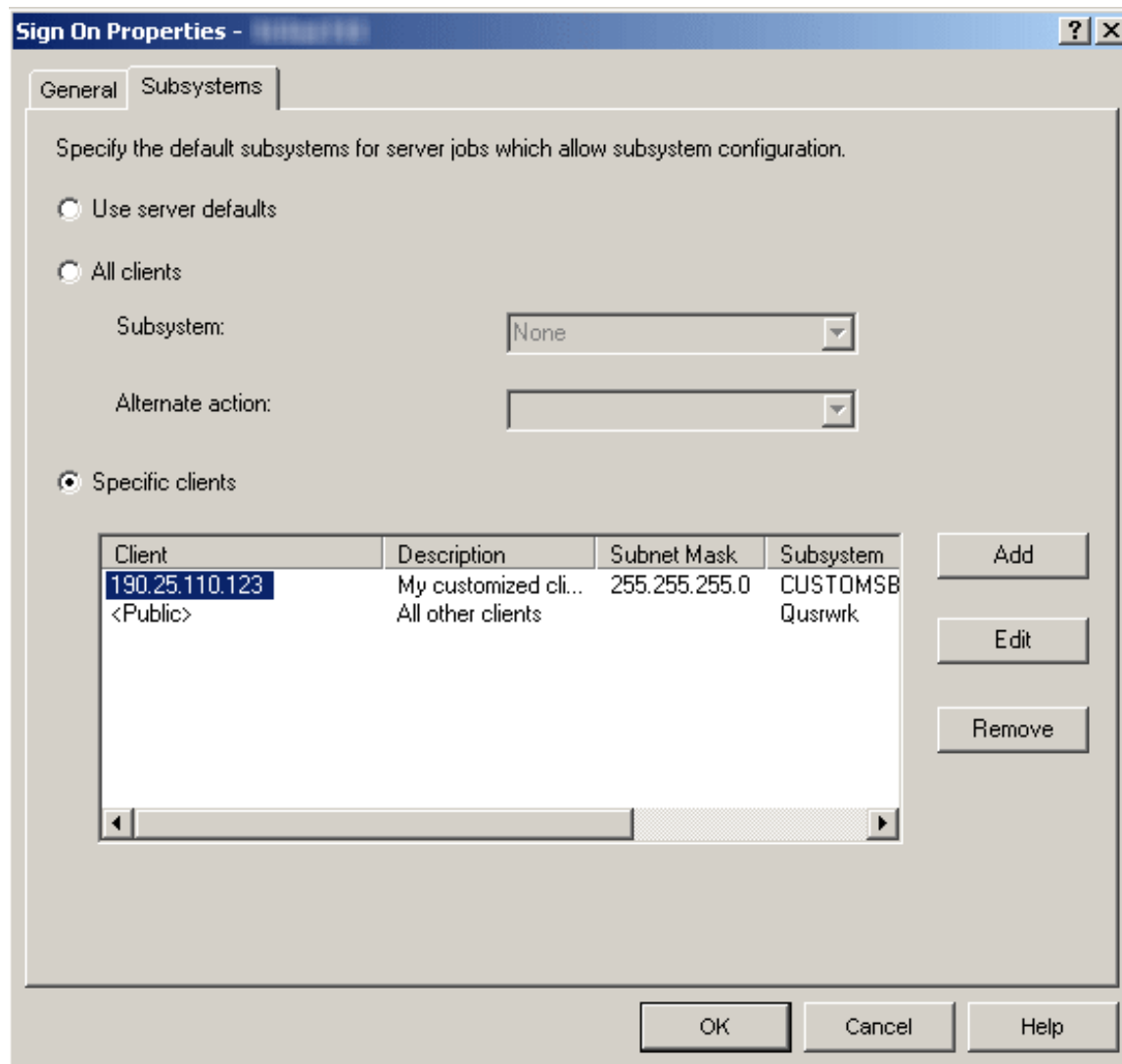


The **Subsystem** list specifies the subsystem you want this server's jobs to run in. If None is selected, the server jobs will perform the Alternate action. You can type in the ten character name of the subsystem you wish to use if it is not contained in the list. If the subsystem is not a system supplied subsystem; see the "User-defined server subsystems" on page 13 information to properly ensure the desired subsystem is setup and running correctly.

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 alternative action would be taken is when the necessary prestart jobs for the given 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 can not 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 in the iSeries Information Center for more information for server jobs, the subsystems in which they run (including server and daemon jobs), and more.

- **Specific clients**

Select this option if you want to set up 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**

Click Add to add a client to the list. This will take you to the Add Client dialog where you can specify the subsystem configuration for a specific client or group of clients. This will bring up a new window, the Add Client dialog.

**Add Client** [?] [X]

Client information

Description:

Client:

IP address:

IP address range:  --

Subnet mask:

Subsystem:  ▼

Alternate action:  ▼

OK Cancel Help

From the Add Client dialog you can specify the subsystem configuration for a specific client or a group of clients.

The **Description** text box specifies a text description of the client(s) that you are configuring.

The **Client** radio button options specifies whether you use an individual IP address or IP address range. An individual IP address should be used for a single client. An IP address range is used for a group of clients. IP address ranges cannot overlap for the selected server.

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 whose hexadecimal representation is X'FFFFFF00' is expressed as 255.255.255.0.

The **Subsystem** list specifies the subsystem you want this server's jobs to run in. If None is selected, the server jobs will perform the Alternate action. You can type in the ten character name of the subsystem you wish to use if it is not contained in the list. If the subsystem is not a system supplied subsystem; see the "User-defined server subsystems" on page 13 information to properly ensure the desired subsystem is setup and running correctly.

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 alternative action would be taken is when the necessary prestart jobs for the given 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 can not 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 in the iSeries Information Center for more information for server jobs, the subsystems in which they run (including server and daemon jobs), and more.

– **Edit**

Click **Edit** to edit a client that you have added to the list. You must select a client from the list before you can edit it. The Edit Client dialog looks identical to the Add Client dialog and has the same configuration options.

– **Remove**

Click **Remove** to delete a client that you have added to the list. You must select the client from the list before you can remove it. You cannot remove the <Public> entry at the end of the list.

After all selections are made, click **OK** to accept and apply the specified server subsystem defaults or click **Cancel** to reject and ignore any changes made to the server subsystem default settings.

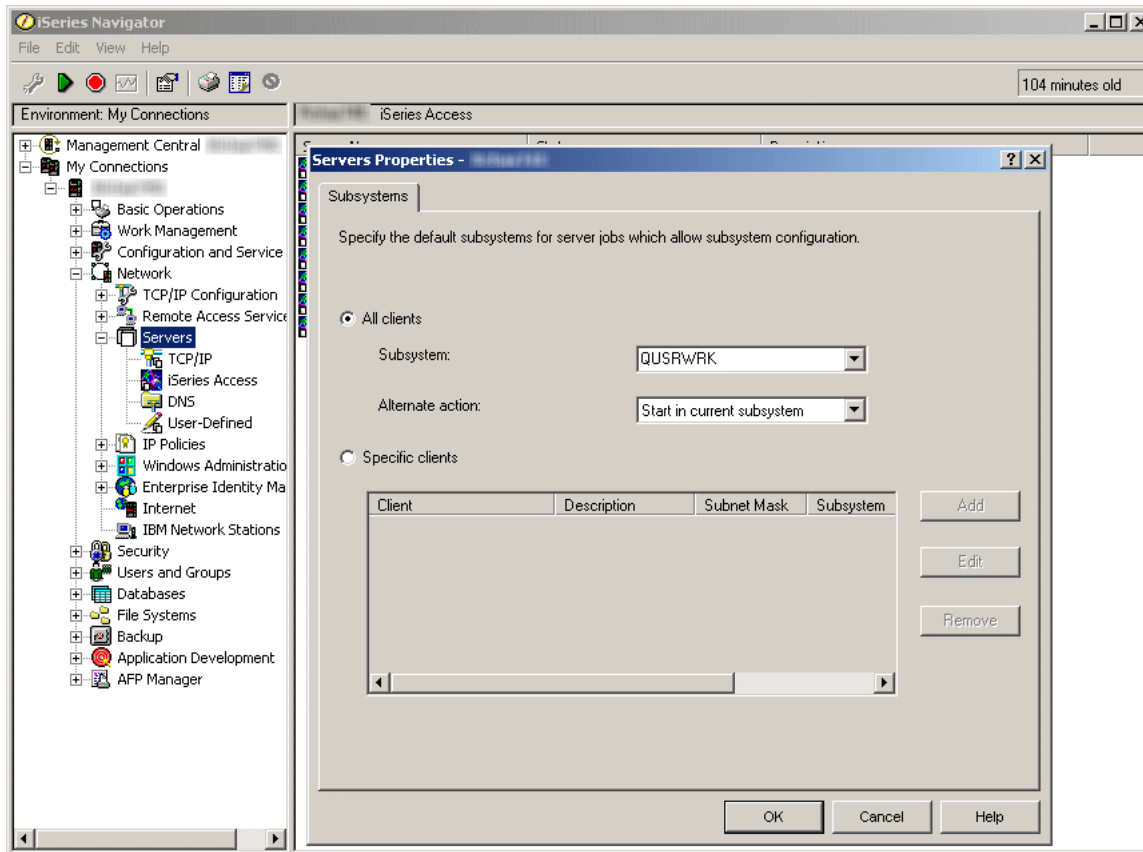
## Server subsystem defaults

Setting the server defaults provides a method to configure in one place the general characteristics for subsystems for all server jobs. The individual servers can then use these defaults. To specify the server defaults, you must use iSeries<sup>(TM)</sup> Navigator; there is no green-screen interface for this function. The following steps show how to set the server subsystem defaults using the iSeries Navigator.

1. Open **iSeries Navigator**.
2. Expand **My Connections**.
3. Expand your **iSeries Server**.
4. Expand **Network**.
5. Select **Servers** and right-click to bring up a menu of options.
6. Open **Properties** by selecting the properties menu option.

The server defaults are specified on the Subsystem properties page for the Servers container.





From the subsystem property page you can specify which default subsystems you want all server 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:

- **All clients**

Specifies that all clients for all servers are to use the same subsystem and alternate action.

The **Subsystem** list specifies the subsystem you want this server's jobs to run in. If None is selected, the server jobs will perform the Alternate action. You can type in the ten character name of the subsystem you wish to use if it is not contained in the list. If the subsystem is not a system supplied subsystem; see the "User-defined server subsystems" on page 13 information to properly ensure the desired subsystem is setup and running correctly.

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 alternative action would be taken is when the necessary prestart jobs for the given 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 can not 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 in the iSeries Information Center for more information for server jobs, the subsystems in which they run (including server and daemon jobs), and more.

- **Specific clients**

Specifies that you want to set up 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 default server subsystem clients.

– **Add**

Click Add to add a client to the list. This will take you to the Add Client dialog where you can specify the subsystem configuration for a specific client or group of clients. This will bring up a new window, the Add Client dialog.

The screenshot shows the 'Add Client' dialog box. The title bar reads 'Add Client' with a help icon and a close button. The dialog is organized into two main areas. The upper area, titled 'Client information', includes a 'Description' text field, a 'Client' section with two radio buttons: 'IP address' (which is selected) and 'IP address range', and a 'Subnet mask' text field containing the value '255.255.255.0'. The lower area contains a 'Subsystem' dropdown menu with 'Qusrwrk' selected and an 'Alternate action' dropdown menu with 'Start in current subsystem' selected. At the bottom of the dialog are three buttons: 'OK', 'Cancel', and 'Help'.

From the Add Client dialog you can specify the subsystem configuration for a specific client or a group of clients.

The **Description** text box specifies a text description of the client(s) that you are configuring.

The **Client** radio button options specify whether you use an individual IP address or IP address range. An individual IP address should be used for a single client. An IP address range is used for a group of clients. IP address ranges cannot overlap for the selected server.

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 whose hexadecimal representation is X'FFFFFF00' is expressed as 255.255.255.0.

The **Subsystem** list specifies the subsystem you want this server's jobs to run in. If None is selected, the server jobs will perform the Alternate action. You can type in the ten character name of the subsystem you wish to use if it is not contained in the list. If the subsystem is not a system supplied subsystem; see the "User-defined server subsystems" on page 13 information to properly ensure the desired subsystem is setup and running correctly.

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 alternative action would be taken is when the necessary prestart jobs for the given 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 can not 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 in the iSeries Information Center for more information for server jobs, the subsystems in which they run (including server and daemon jobs), and more.

– **Edit**

Click **Edit** to edit a client that you have added to the list. You must select a client from the list before you can edit it. The Edit Client dialog looks identical to the Add Client dialog and has the same configuration options.

– **Remove**

Click **Remove** to delete a client that you have added to the list. You must select the client from the list before you can remove it. You cannot remove the <Public> entry at the end of the list.

After all selections are made, click **OK** to accept and apply the specified server subsystem defaults or click **Cancel** to reject and ignore any changes made to the server subsystem default settings.

## User-defined server subsystems

The following article describes how to create a new user-defined server subsystem that is based off of the QUSRWRK system supplied subsystem. User-defined subsystems can be used for “Server subsystem configuration” on page 3.

One of the first planning steps is to determine the naming convention you will use for your subsystems. The naming convention can be something simple, something more reflective of your users (INVENTORY, PGMR, ORDERENT), or reflective of the geographic location in which your users reside (EAST, CENTRAL, WEST). Decide upon a naming convention that will be meaningful for your system administration.

The steps below are described as if the subsystem was created interactively. You can use a CL program to create your subsystems so you can easily recreate your configurations for recovery purposes.

1. Create a library to store your subsystem configuration objects in. In this example, we will use SBSLIB.  
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.

The following example creates a class MYSBS in the SBSLIB library that has a priority of 20, a time slice of 2000 milliseconds, and has a default wait time of 30 seconds.

```
CRTCLS SBSLIB/MYSBS RUNPTY(20) TIMESLICE(2000) DFTWAIT(30)
TEXT('Custom Subsystem Class')
```

For more information on creating a class see the Create Class (CRTCLS) command description.

**Note:**

You can create a single class to use for all of your subsystems, or you can create a class for each subsystem. Which you choose depends upon whether you want to customize some of the performance settings for particular subsystems. IBM<sup>(R)</sup>-supplied subsystems are shipped with a class created for each subsystem, and the name of the class is 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 since the default for the class (CLS) parameter is \*SBSD, which means the class name has the same name as the subsystem description.

The following example demonstrates why you may want different classes for different subsystems. If you want to have your critical users have a higher run priority than the remainder of your users, you could accomplish this by having those critical users run in their own subsystem, and the class used for that subsystem could specify a higher run priority. Remember for run priority that a lower number gives a higher priority.

3. Create the subsystem description. You will repeat this step for each subsystem you need to define. More information on how to create a subsystem description can be found in the iSeries Information Center.

The following example creates a subsystem description with attributes identical to those of QUSRWRK. Job queue, routing entries, and prestart job information will have to be added manually at a later time.

```
CRTSBSD SBSD(SBSLIB/MYSBS) POOLS((1 *BASE)) TEXT('Custom Server Subsystem')
```

After a subsystem description is created, use the Display Subsystem Description (DSPSBSD) command to display the subsystem description information.

As an alternative to creating a subsystem description from scratch, you can copy an existing subsystem description using the Create Duplicate Object (CRTDUPOBJ) command. The following example creates a subsystem description with attributes identical to those of QUSRWRK. This will also copy job queue, routing entry, and prestart job information from QUSRWRK. Depending on the requirements for your subsystem, these values may have to be changed.

```
CRTDUPOBJ OBJ(QUSRWRK) FROMLIB(QSYS) OBJTYPE(*SBSD) TOLIB(SBSLIB)  
NEWOBJ(MYSBS)
```

This report assumes that a subsystem description was created from scratch

#### Note:

The storage pools used are specified on the subsystem description. If you want to isolate a set of users to a pool of their own, you can do this by specifying a pool ID with a storage size and activity level specifically for the subsystem, rather than using the \*BASE pool. You could also define a shared pool with the Change Shared Storage Pool (CHGSHRPOOL) command and specify that shared pool on the subsystem description.

The subsystem description also defines the number of jobs allowed to run in the subsystem. The following parameters are available the Create Subsystem Description (CRTSBSD) command, but add complexity that may make it much easier to unintentionally prevent users from accessing the subsystem, thus the recommendation to not use these parameters.

**Activity Level (MAXACT):** The activity level determines the maximum number of threads that can be actively running at one time. More threads than this can be active within the subsystem, but the activity level determines the number that can be actively running at any given point in time.

**Maximum Jobs (MAXJOBS):** The maximum number of jobs parameter determines the number of jobs running in the subsystem. This value cannot be exceeded. Any attempt to start additional jobs when the maximum number of jobs is already running will fail.

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 SBSD(SBSLIB/MYSBS) JOBQ(SBSLIB/MYSBS) MAXACT(*NOMAX)
```

For more information on each command see the Create Job Queue (CRTJOBQ) command description and the Add Job Queue Entry (ADDJOBQE) command description. The iSeries Information Center also has information on how a job queue works.

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 on a user-defined subsystem is to make them identical to the routing entries used on QUSRWRK.

```
ADDRTGE SBSD(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.

See the Add Routing Entry (ADDRTGE) command description for more information.

**Note:**

**Maximum Activity Level (MAXACT):** The maximum activity level 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 ADDRTGE command.

You can get quite complex with routing entries. If you need to investigate the options available refer to the Work Management book.

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 on a user-defined subsystem be identical to the prestart job entries used on QUSRWRK.

```
ADDPJE SBSD(SBSLIB/MYSBS) PGM(QSYS/QZSOSIGN) INLJOBS(50) THRESHOLD(4)
JOB(QZSOSIGN) JOBD(QSYS/QZBSJOB) CLS(QGPL/QCASERVR) STRJOBS(*YES)
```

Do not use the defaults on the command. For information on balancing the initial number of jobs, threshold, additional number of jobs parameters and other important prestart job settings, see the tuning prestart job entries information. See the Add Prestart Job Entry (ADDPJE) command description for more information on how to add a prestart job entry to a subsystem description.

**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. See “Server subsystem configuration” on page 3 information for details on alternative actions for subsystems.

The following parameters are used to determine the number of prestart jobs to initially run and what to do when the current pool of initial prestart jobs is in use.

- Initial number of jobs (INLJOBS): Specifies the initial number of prestart jobs that are started when the subsystem specified on the Subsystem description (SBSD) parameter is started. The value of this parameter must be less than or equal to the value of the Maximum number of jobs (MAXJOBS) parameter. The value of this parameter must be greater than or equal to the value of the Threshold (THRESHOLD) parameter. The current workload is not the same as the initial number of jobs. A good rule of thumb is to set the initial number of jobs to the expected workload plus the threshold.
- Threshold (THRESHOLD): Specifies when additional prestart jobs are started. When the pool of available jobs (jobs available to service requests) is reduced below this number, more jobs (specified on the Additional number of jobs (ADLJOBS) parameter) are started and added to the available pool. The value of this parameter must be less than or equal to the value specified on the Initial number of jobs (INLJOBS) parameter. The threshold should reflect the system workload and the amount of new work coming in once all available prestart jobs unavailable.
- Additional number of jobs (ADLJOBS): Specifies the additional number of prestart jobs that are started when the number of prestart jobs drops below the value specified on the Threshold (THRESHOLD) parameter. The value specified on this parameter must be less than the value specified on the Maximum number of jobs (MAXJOBS) parameter.

For more information on prestart jobs, see prestart jobs, using prestart jobs, and use of prestart jobs information in the Work Management section of the iSeries Information Center.

Once the subsystem descriptions have been created, use the Display Subsystem Description (DSPSBSD) command to display the various attributes of the subsystem and verify that the set up has been done correctly.

Start the subsystems using the Start Subsystem (STRSBS) command. To verify that work is being processed by the subsystem, “Server subsystem configuration” on page 3 to use a user-defined subsystem and make sure appropriate clients are up and running in the subsystem. The Work with Subsystem Jobs (WRKSBSJOB) command can help verify that jobs are running in the user-defined subsystem.

Any user-defined subsystem will have to be started before any servers that use the user-defined subsystem are started. The best way to ensure that a user-defined subsystem is available for a server, is to start the subsystem before TCP/IP starts. There are two ways to do this. This first is changing the IPL start-up program. This has the benefit of automatically starting subsystems during an IPL, which avoids having to manually start them each time the system is restarted or started from the restricted state. A second way to control subsystem start-up is to change the STRTCP IPL attribute to \*NO and then to manage starting the subsystems, TCP/IP, and server jobs in the right order in the system startup program. See the Start TCP/IP (STRTCP) command description for more information including the IPL autostart information. There is more information on how to start the server in the iSeries Information Center.

---

## Managing server jobs

With the ability to customize server subsystems comes a desire to have better control over the server jobs that run in a given subsystem. iSeries<sup>(TM)</sup> Navigator provides many different interfaces for locating and managing server jobs. After a server job is located, there are a variety of functions that can be used to control and manage these jobs.

This experience report includes the following information:

### “All server jobs” on page 17

This section describes how to find all server jobs using iSeries Navigator.

**“Server jobs by server” on page 18**

This section describes how to find server jobs for a given server using iSeries Navigator.

**“Server jobs by user” on page 20**

This section describes how to find server jobs based on the user using iSeries Navigator.

**“Ended server jobs” on page 22**

This section describes how to find server jobs that have ended using iSeries Navigator

**“Server job table” on page 25**

This section shows a list of server jobs that run in QSYSWRK

**“Server job details and properties” on page 25**

This section describes the server job details and properties that can be managed using iSeries Navigator.

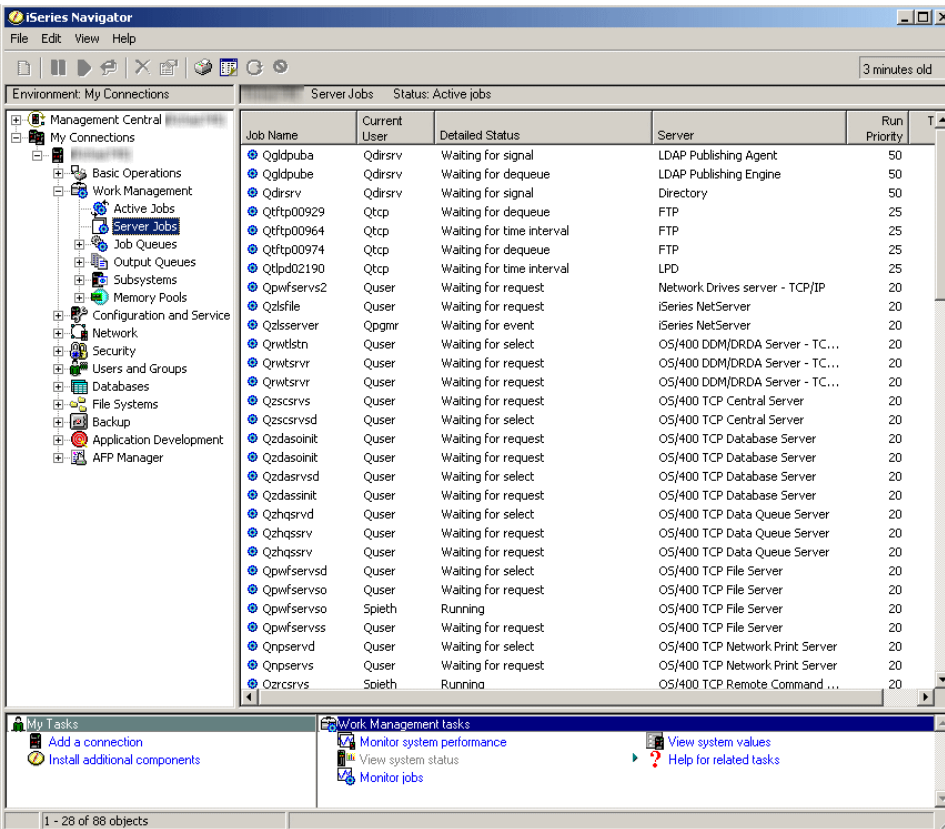
## All server jobs

The iSeries<sup>(TM)</sup> Navigator allows you to look at all server jobs that are not on an output queue. It can be beneficial to see all server jobs, regardless of which server they belong to. This aids in finding a specific server job if the owning server is unknown.

While there are green screen interfaces for server jobs, the following information uses iSeries Navigator. For more information on server jobs and green screen interfaces to them, see the identifying server jobs on the iSeries server information located in the iSeries Information Center.

The following steps show how to view all server jobs using the iSeries Navigator.

1. Open **iSeries Navigator**.
2. Expand **My Connections**.
3. Expand your **iSeries Server**.
4. Expand **Work Management**.
5. Select **Server Jobs**. A list of all server jobs is presented in the right pane.



6. Find the server job you want to manage.
7. Right-click the **Job Name**
8. Select a **Menu Option**

See “Server job details and properties” on page 25 for more information about each menu option that is available for managing a server job.

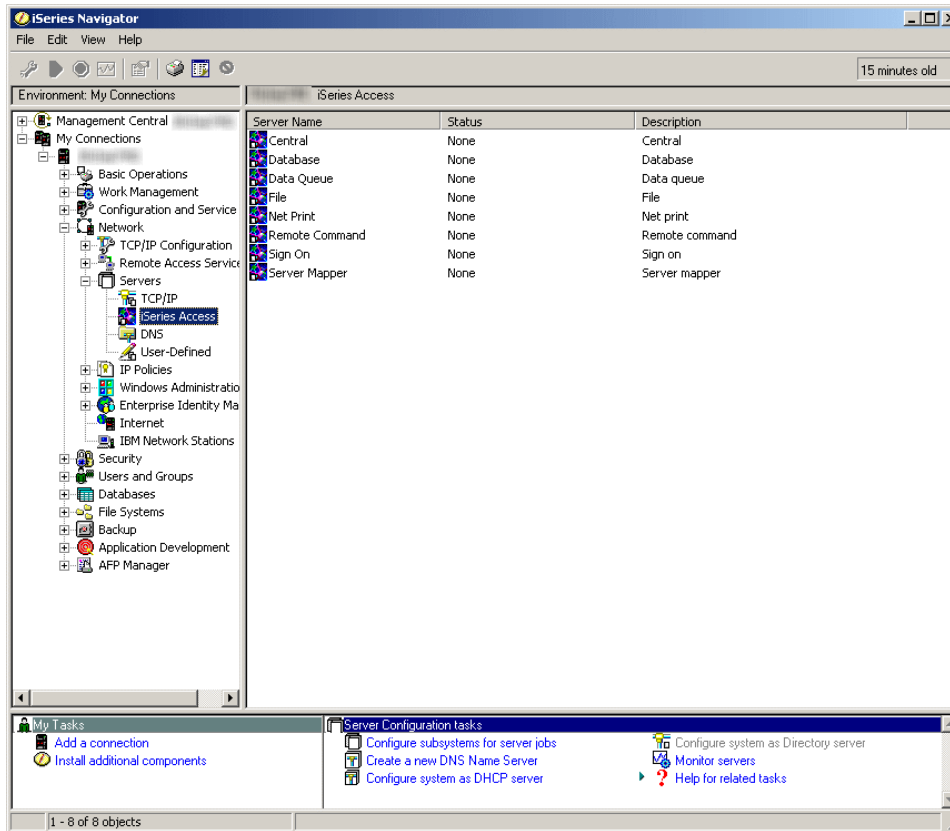
## Server jobs by server

The following steps show how to view all server jobs for a specific server using the iSeries<sup>(TM)</sup> Navigator.

1. Open **iSeries Navigator**.
2. Expand **My Connections**.
3. Expand your **iSeries Server**.
4. Expand **Network**.
5. Expand **Servers**.
6. Select **TCP/IP** or **iSeries Access**. The specific servers will show in the right pane.

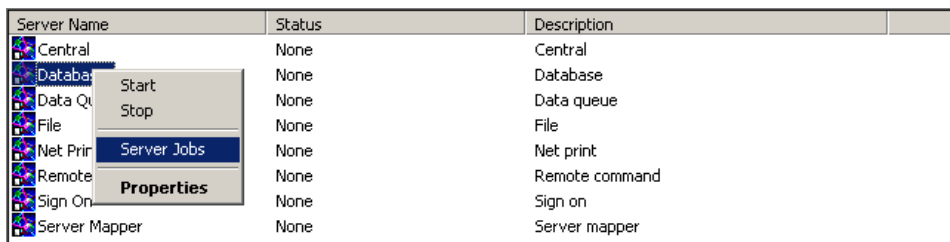
The option you select is determined by the specific server you want to manage server jobs for. **TCP/IP** servers include the DDM and iSeries NetServer servers. **iSeries Access** servers include the Central, Database, Data Queue, Files, Net Print, Remote Command, and Sign On servers.





The remaining steps use an iSeries Access server.

7. Right-click the **Server Name** then select **Server Jobs**.



8. Another window opens from which server jobs can be managed.

Job Name	Current User	Detailed Status	Server	Run Priority	Thread Count
Qzdassinit	Quser	Waiting for request	OS/400 TCP Database Server	20	1
Qzdasrvsd	Quser	Waiting for select	OS/400 TCP Database Server	20	1
Qzdasoinit	Quser	Waiting for request	OS/400 TCP Database Server	20	1
Qzdasoinit	Quser	Waiting for request	OS/400 TCP Database Server	20	1

From this window, server jobs can be managed and information customized just as it can be from the right pane on the main window of the iSeries Navigator.

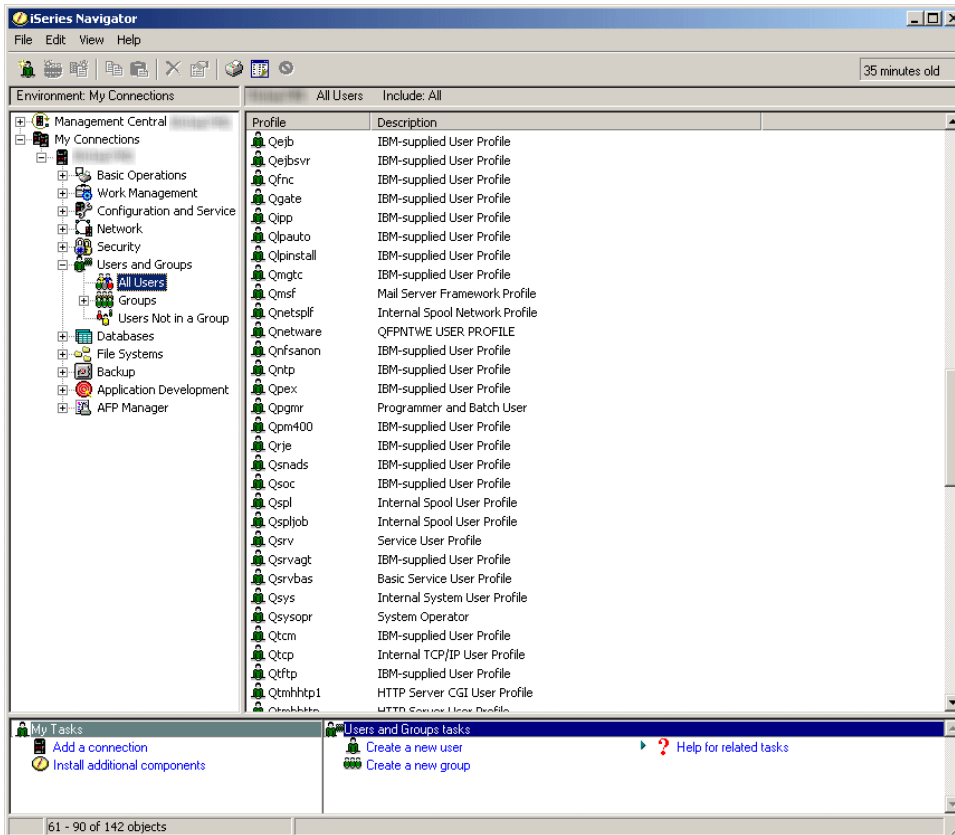
9. **Find** the server job you want to manage.
10. Right-click the **Job Name**
11. Select a **Menu Option**

See “Server job details and properties” on page 25 for more information about each menu option that is available for managing a server job.

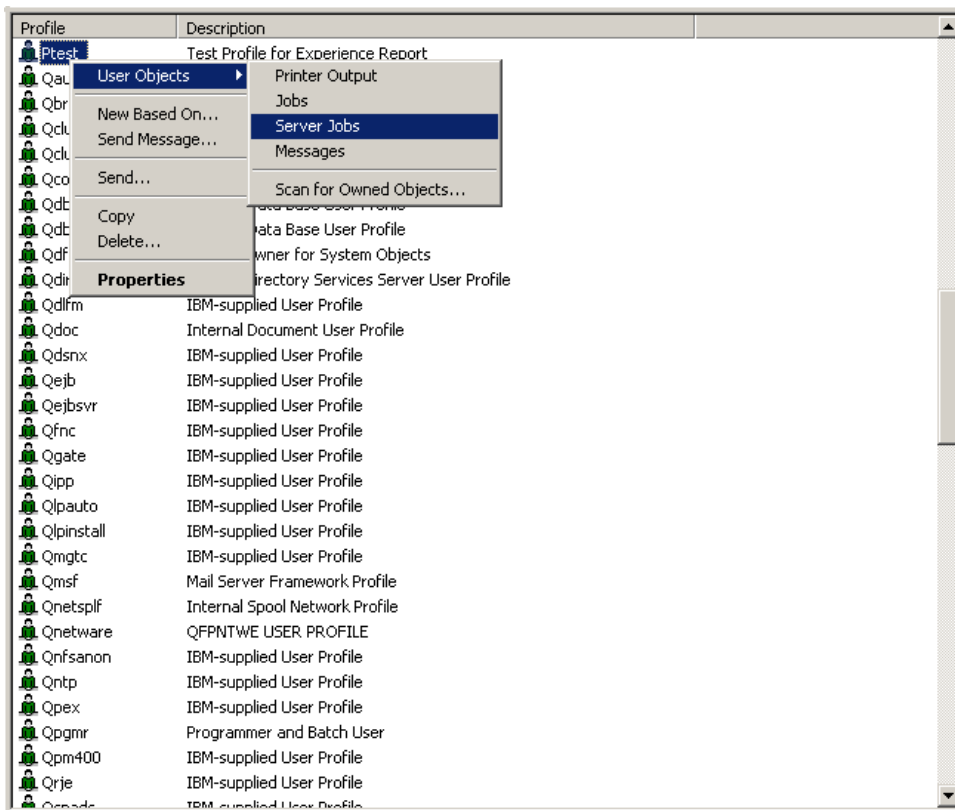
## Server jobs by user

The following steps show how to view all server jobs for a specific user using the iSeries<sup>(TM)</sup> Navigator. The server jobs retrieved for a given user profile will be those server jobs that are actually running under the given user’s profile. This is not necessarily the same user profile that the server job was initiated under. Keep this in mind when determining which user profile to select to see server jobs that are running for that user.

1. Open **iSeries Navigator**.
2. Expand **My Connections**.
3. Expand your **iSeries Server**.
4. Expand **Users and Groups**.
5. Select **All Users**.



- Right click the **Profile** of the user for which you want to see server jobs and select **User Objects** then **Server Jobs**.



- Another window opens from which server jobs for this user can be managed.

Job Name	Current User	Detailed Status	Server	Run Priority	Thread Count
Qpwfservso	Ptest	Running	OS/400 TCP File Server	20	1
Qpwfservso	Ptest	Running	OS/400 TCP File Server	20	1
Qzrcsrvs	Ptest	Waiting for time interval	OS/400 TCP Remote Command ...	20	1
Qzdasoinit	Ptest	Waiting for time interval	OS/400 TCP Database Server	20	1
Qzrcsrvs	Ptest	Running	OS/400 TCP Remote Command ...	20	1
Qqyserver	Ptest	Waiting for dequeue	OS/400 Open List Server	20	1

From this window, server jobs can be managed and information customized just as it can be from the right pane on the main window of the iSeries Navigator.

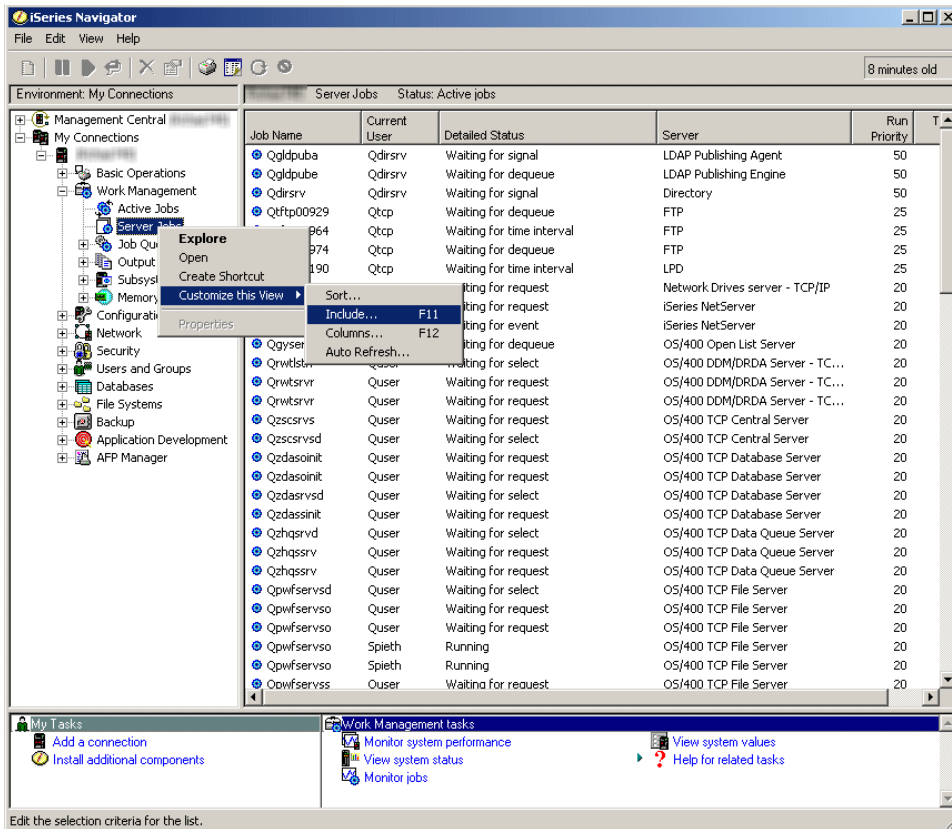
8. **Find** the server job you want to manage.
9. Right-click the **Job Name**
10. Select a **Menu Option**

See “Server job details and properties” on page 25 for more information about each menu option that is available for managing a server job.

## Ended server jobs

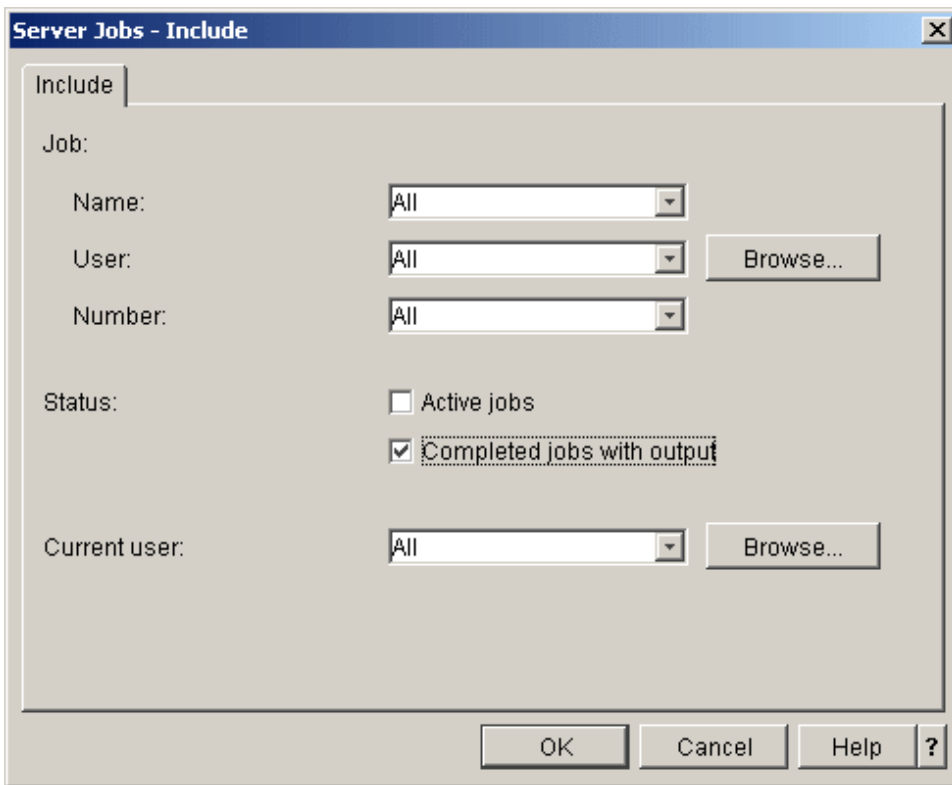
When a server job that has spooled output has ended, the job remains on the system and can be viewed. The following steps describe how to find ended server jobs with spooled output using the iSeries<sup>(TM)</sup> Navigator.

1. Open **iSeries Navigator**.
2. Expand **My Connections**.
3. Expand your **iSeries Server**.
4. Expand **Work Management**.
5. Right-click **Server Jobs** and select **Customize this View** then select **Include**.



The Server Jobs -Include (see 24) dialog box is displayed.

6. Select **Completed Jobs with Output** and click **OK**.



The server job list will be changed to show only completed jobs with output.

7. **Find** the completed server job you want to manage.

8. Right-click the **Job Name**
9. Click a **Menu Option**

See “Server job details and properties” on page 25 for more information about each menu option that is available for managing a server job.

### Server Jobs - Include

Use the Server Jobs - Include dialog to limit the list of server jobs displayed in the iSeries Navigator window to those that meet the criteria you specify. The following will describe each section of this dialog.

- **Job**

Specify the following criteria to be used to limit the list of jobs displayed in the iSeries Navigator window.

- **Name**

Names of jobs to include in the list. Select from the following values:

- **All** - All job names are allowed. The job name is not used to subset the list.
- **Specific name** - Jobs currently running under a specific job name are included. For example, if you enter the name QZDASOINIT, only jobs with this name will show.
- **Generic name** - Jobs beginning with the characters specified are included. Enter one or more characters followed by a trailing asterisk (\*). The asterisk cannot be entered between characters.

- **User**

Users whose jobs should be included in the list. Possible values are:

- **All** - All user names are allowed. The user name is not used to subset the list.
- **Current signed on user** - Jobs belonging to the current user that is signed on are included. This option will look for server jobs with user profiles that match the profile for the current signed on user as part of the qualified server job name. In many instances this option will not apply for server jobs.
- **Specific name** - Jobs currently running under a specific user are included. Enter a valid user profile name, or select Browse... to select from a list of users.
- **Generic name** - User names beginning with the characters specified are included. Enter one or more characters followed by a trailing asterisk (\*). The asterisk cannot be entered between characters.
- **Browse** - Click Browse to bring up the Browse User dialog. Select a **user** from the list and click **OK**.

- **Number**

System-generated job numbers for jobs to include in the list. Possible values are:

- **All** - All job numbers are allowed. The job number is not used to subset the list.
- **Specific number** - Jobs with specific job number are shown. Enter a six-digit job number from 000000-999999.

- **Status**

Jobs of the specified status are included in the list. The status of a job tells where the job is in the system in relation to when it will run. Possible options are:

- **Active Job** - Select to include jobs that have started running but have not completed running.
- **Completed jobs with output** - Select to include jobs that have finished running and have spooled output.

- **Current user**

Jobs running under the current user are included in the list. Possible options are:

- **All** - Jobs for all users are included.
- **Specific name** - Jobs currently running under a specific user are included. Specify a valid user profile name, or select Browse... to select from a list of users.

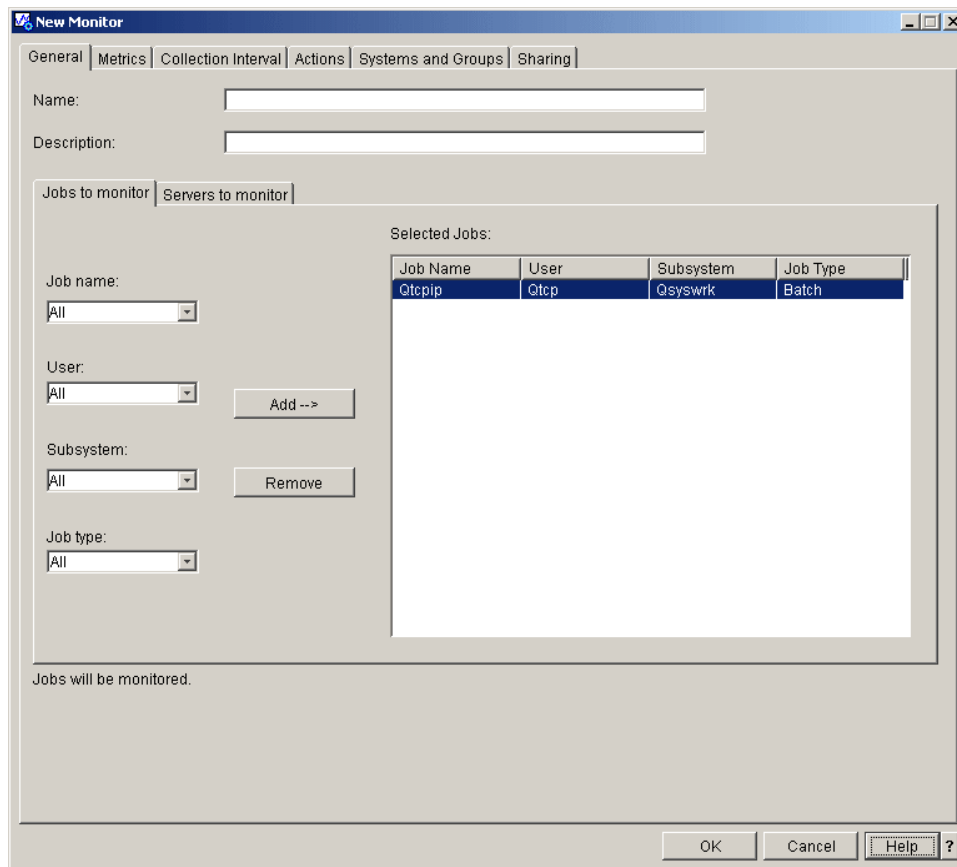


- Call Stack (see 28)
- Library List (see 28)
- Locked Objects (see 29)
- Opened Files (see 30)
- Threads (see 30)
- Transactions (see 31)
- Elapsed Performance Statistics (see 32)
- Last SQL Statement (see 34)
- Hold (see 35)
- Delete/End (see 36)
- Properties (see 38)

## Monitor

Select **Monitor**. The Monitor dialog opens in a new window.

You can use a job monitor to monitor a job or list of jobs based on job name, job user, job type, subsystem, or server type. For example, you might want to monitor a job's CPU usage, job status, or job log messages. The job monitor allows you to define commands to run when a specified threshold is met. The scenario: job monitor for CPU utilization in the iSeries Information Center gives a good example of how to properly establish and configure a job monitor.



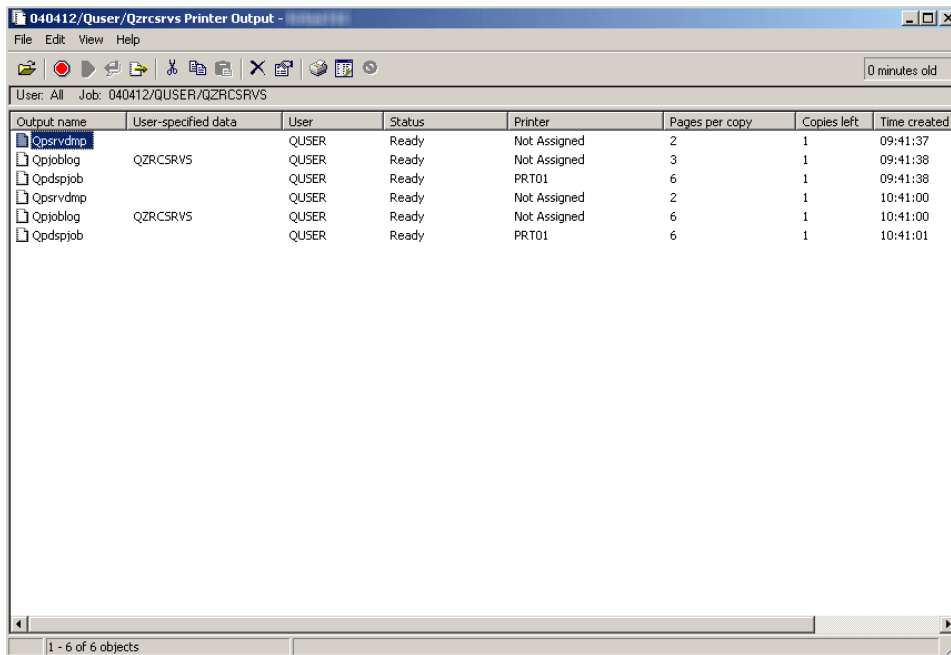
The General page for New Monitor or Monitor Properties allows you to view and change general information about the monitor. The general information includes the name of the monitor, a brief description of the monitor, and the jobs and servers to monitor. See the create a new monitor information



in the iSeries Information Center for more information on how to create a new monitor. Additionally see the Management Central topic in the iSeries Information Center for more information.

## Printer Output

Select **Printer Output**. A new window opens from which the job's printer output can be viewed.



The screenshot shows a window titled "040412/Quser/Qzrcsrvs Printer Output" with a menu bar (File, Edit, View, Help) and a toolbar. Below the toolbar, it says "User: All Job: 040412/QUSER/QZRCRSRV5". The main area contains a table with the following data:

Output name	User-specified data	User	Status	Printer	Pages per copy	Copies left	Time created
Qpsrvdmp		QUSER	Ready	Not Assigned	2	1	09:41:37
Qpjoblog	QZRCRSRV5	QUSER	Ready	Not Assigned	3	1	09:41:38
Qpdsplib		QUSER	Ready	PRT01	6	1	09:41:38
Qpsrvdmp		QUSER	Ready	Not Assigned	2	1	10:41:00
Qpjoblog	QZRCRSRV5	QUSER	Ready	Not Assigned	6	1	10:41:00
Qpdsplib		QUSER	Ready	PRT01	6	1	10:41:01

At the bottom of the window, it says "1 - 6 of 6 objects".

The following can be used to access and customize the displayed information:

- Display, Open, Hold, Release, Print Next, Send, Cut, Copy, Move, or Delete printer output.
- Change printer output properties
- Tailor the printer output display
- Drag and drop printer output

## Job Log

Select **Job Log**. A new window opens from which the job's job log can be viewed.

Message ID	Message Text	Sent	Message Type	Severity
CPF1124	Job 039889/QTCP/QTCPIP started on 10/02/03 at 10:08:12 ...	10/2/03 10:08:12 AM	Information	0
CPI1125	Job 039889/QTCP/QTCPIP submitted.	10/2/03 10:08:12 AM	Information	0
CPC1221	Job 039893/QTCP/QTCPPMONITR submitted to job queue QS...	10/2/03 10:08:19 AM	Information	0
TCP1A95	Job 039889/QTCP/QTCPIP started.	10/2/03 10:08:21 AM	Completion	0
TCP1B07	User QTCP in job 039889/QTCP/QTCPIP started TCP/IP *LO...	10/2/03 10:08:22 AM	Information	0
CPC1221	Job 039894/QTCP/QTOCPP submitted to job queue Q5YSNO...	10/2/03 10:08:22 AM	Completion	0
TCP1B31	Status of *LIN TRNLINE is 'VARIED ON'.	10/2/03 10:08:22 AM	Information	0
CPD2645	Line TRNLINE already varied on.	10/2/03 10:08:22 AM	Diagnostic	40
CPC2609	Vary on completed for controller TRNLINET.	10/2/03 10:08:22 AM	Completion	0
CPC2605	Vary on completed for device TRNLITCP.	10/2/03 10:08:22 AM	Completion	0
TCP1B31	Status of *CTL TRNLINET is 'ACTIVE'.	10/2/03 10:08:22 AM	Information	0
TCP1B31	Status of *DEV TRNLITCP is 'VARIED ON'.	10/2/03 10:08:22 AM	Information	0
TCP1B33	Activation request '3B032574DE00TRNLITCP' issued.	10/2/03 10:08:22 AM	Information	0
TCP1B34	Activation request '3B032574DE00TRNLITCP' complete. Cod...	10/2/03 10:08:23 AM	Information	0
TCP1B03	User QTCP started IP interface 9.5.6.5 on TRNLINE.	10/2/03 10:08:26 AM	Completion	0
TCP1B03	User QTCP started IP interface 9.5.13.88 on TRNLINE.	10/2/03 10:08:26 AM	Completion	0

The following can be used to access and customize the displayed information:

- Tailor the job log display
- Sort the job log display

### Call Stack

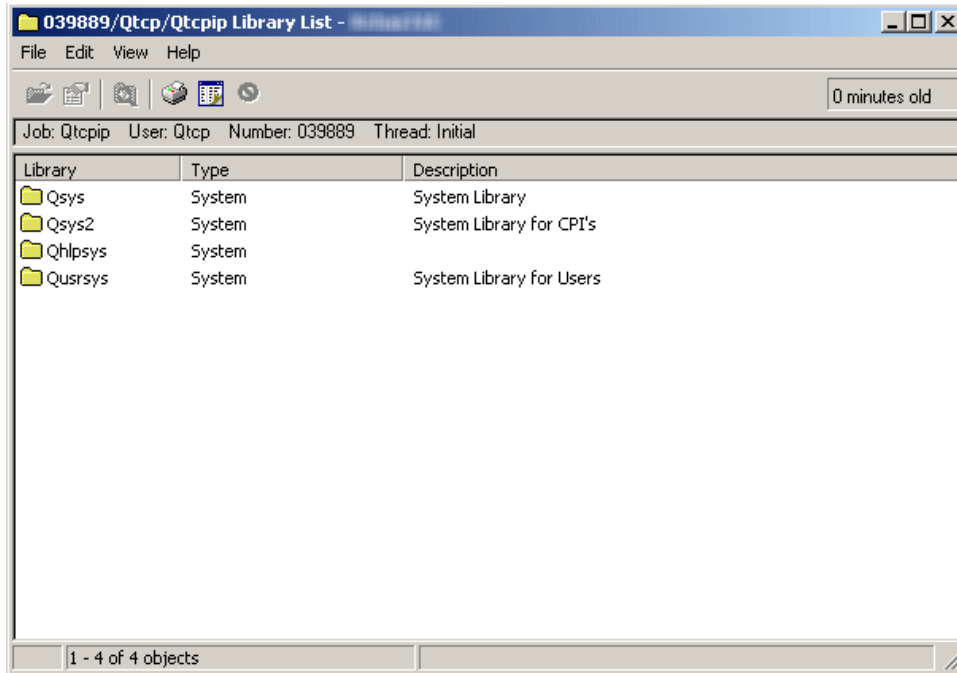
Select **Details** then **Call Stack**. A new window opens from which the job's call stack can be viewed.

Call Level	Procedure	Module	Program	Statements
5			Qtoctpi	
4	toQtcpipJob5PI	Totcpjbspi	Qtoctcpip	2504
3	startTasks__FR18QtcpipReqContext_T	Qtoctcpip	Qtoctcpip	34
2	main	Qtoctcpip	Qtoctcpip	38
1	_CXX_PEP__Fv	Qtoctcpip	Qtoctcpip	6

The call stack can be printed but it can not be modified.

### Library List

Select **Details** then **Library List**. A new window opens from which the job's library list can be viewed.

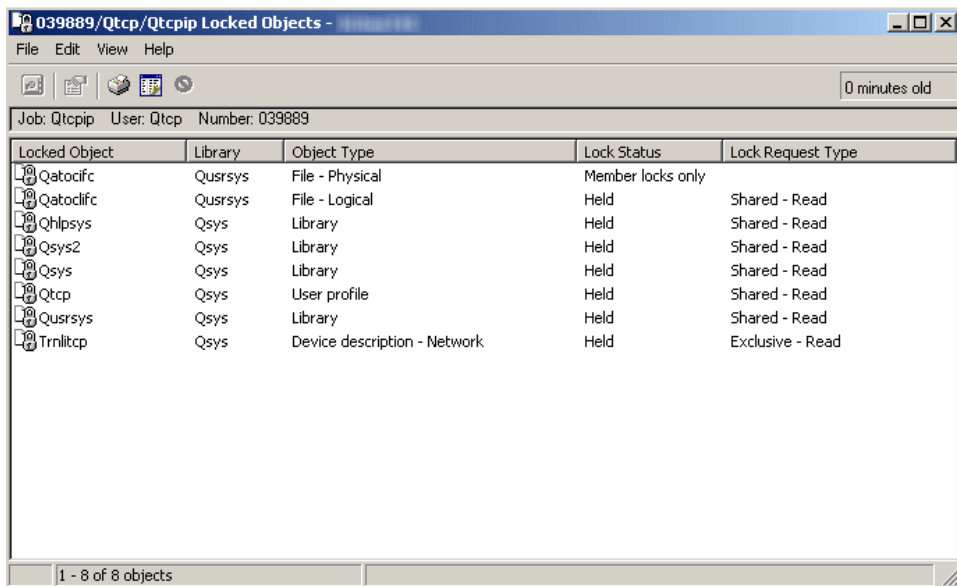


Select a library from the library list. After selecting a library you can do the following:

- Open the library
- Display the library properties
- Search the library list

### Locked Objects

Select **Details** then **Locked Objects**. A new window opens from which objects locked by the job can be viewed.



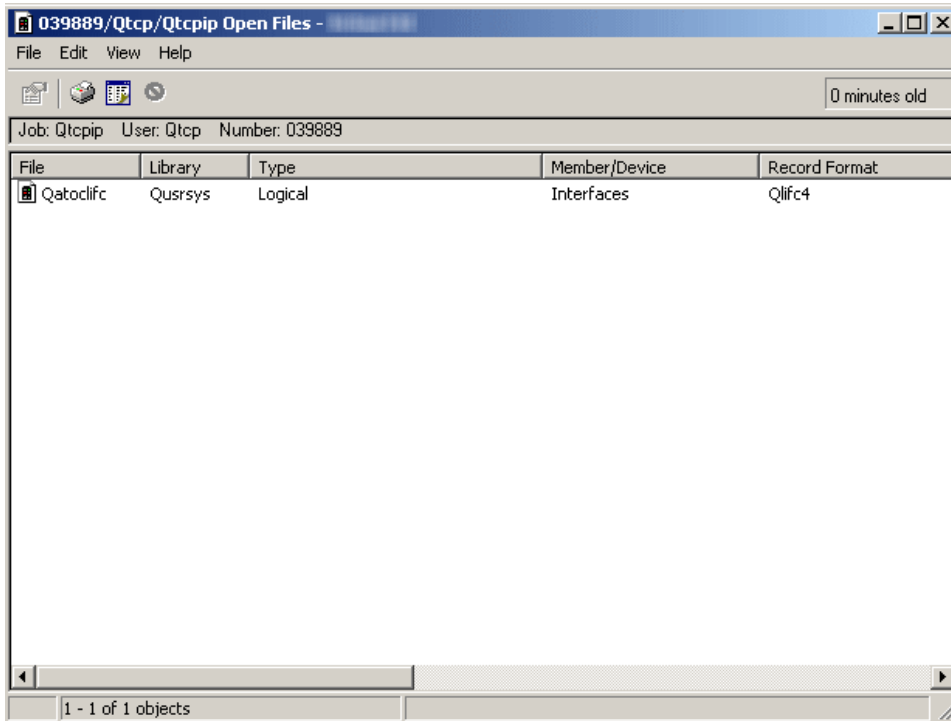
Select a locked object from the list. After selecting a locked object you can do the following:

- View the lock holders
- View the locked members

- View the lock properties

## Opened Files

Select **Details** then **Opened Files**. A new window opens from which files opened by the job can be viewed.



The screenshot shows a window titled "039889/Qtcp/Qtcpip Open Files" with a menu bar (File, Edit, View, Help) and a toolbar. Below the toolbar, it displays "Job: Qtcpip User: Qtcp Number: 039889" and "0 minutes old". The main area contains a table with the following data:

File	Library	Type	Member/Device	Record Format
Qatoclfc	Qusrsys	Logical	Interfaces	Qlfc4

At the bottom of the window, a status bar indicates "1 - 1 of 1 objects".

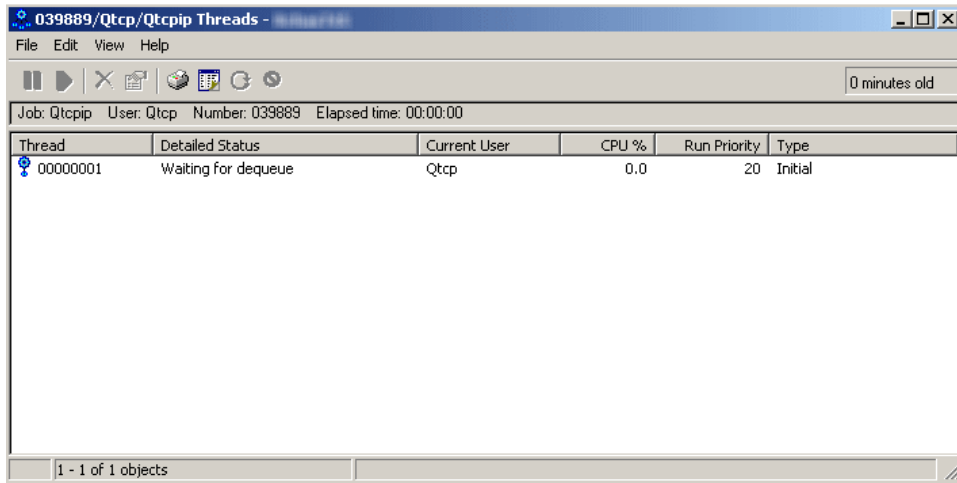
Select an opened file from the list. After selecting an opened file you can do the following:

- Display the file properties

## Threads

Select **Details** then **Threads**. A new window opens from which job threads can be viewed.

A thread can be considered an independent unit of dispatchable work within a job. See threads information from the Work Management topic in the iSeries Information Center for more information.



Select a thread from the list. After selecting a thread you can do the following:

- Hold the thread
- Release the thread
- Delete/End the thread
- View the thread's properties
- View the thread's call stack, library list, locked objects, transactions, and elapsed performance statistics
- Reset thread statistics

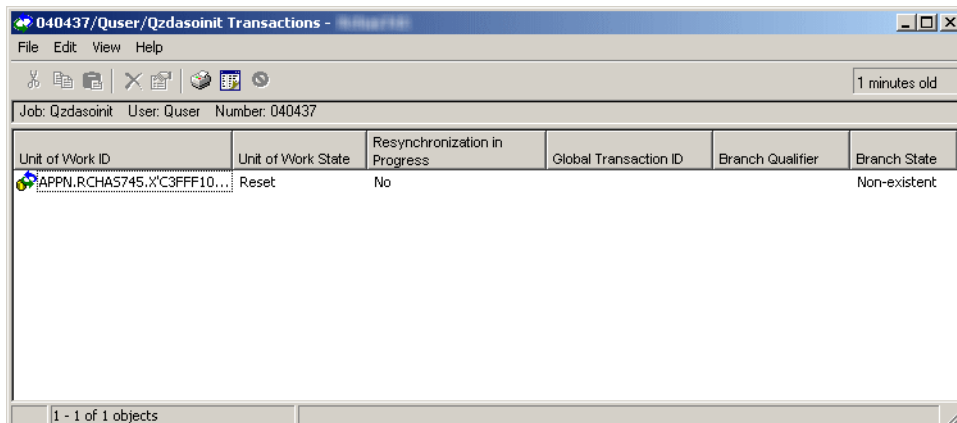
## Transactions

Select **Details** then **Transactions**. A new window opens from which job transactions can be viewed.

A transaction is defined as a group of individual changes to objects on the system that should appear as a single atomic change to the user. A transaction is sometimes called a logical unit of work (LUW). Commitment control is used to ensure that either the entire group of individual changes occur on all systems that participate or that none of the changes occur.

iSeries server supports two different types of transactions: Global transactions and Database transactions. A Global transaction may contain changes both outside and within DB2 UDB for iSeries. A Database Transaction contains only DB2<sup>(R)</sup> UDB for iSeries<sup>(TM)</sup> changes.

For more information about Commitment Control and transactions, see commitment control in the iSeries Information Center

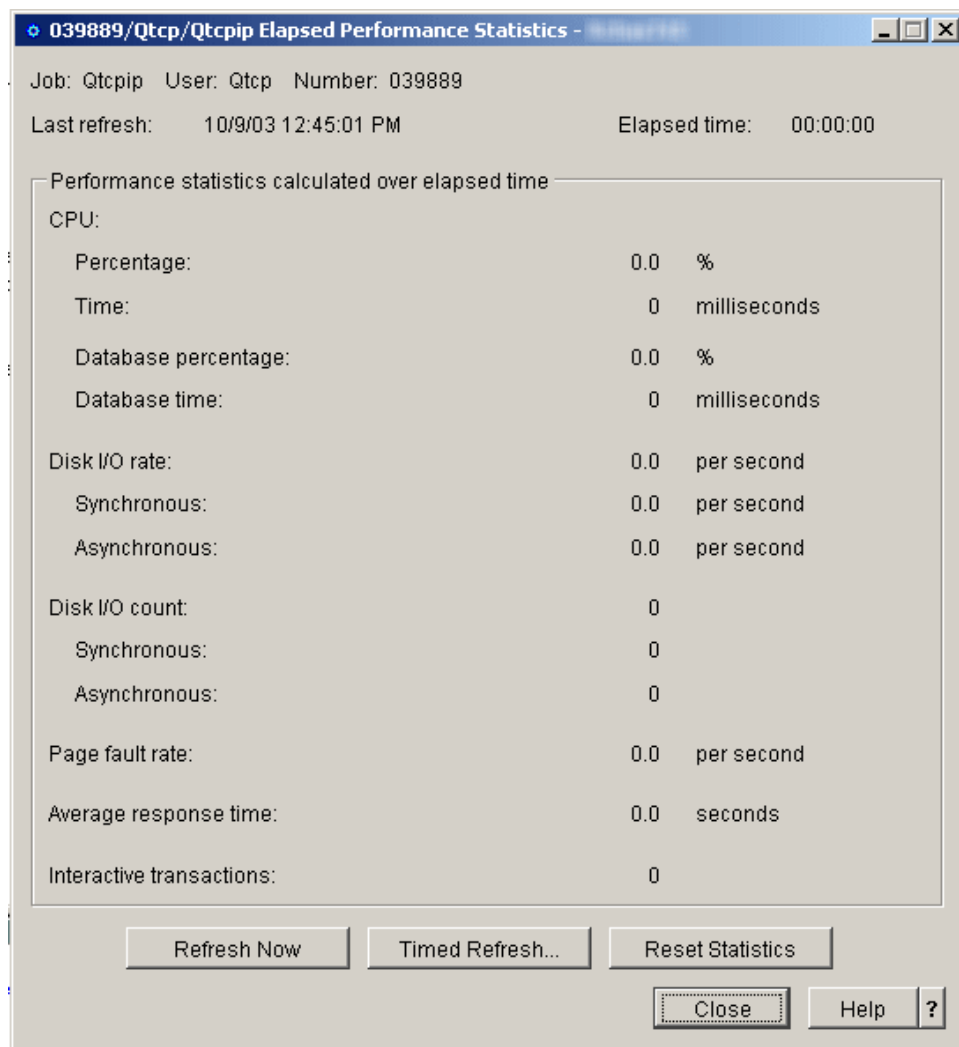


Select a Unit of Work ID (transaction). After selecting a transaction you can do the following:

- View transaction properties
- View the transaction's jobs
- View resource statistics for the transaction
- Force a commit of the transaction
- Force a rollback of the transaction
- Cancel resynchronization for the transaction

### Elapsed Performance Statistics

Select **Details** then **Elapsed Performance Statistics**. The elapsed performance statistics dialog opens in a new window.



Use the **Elapsed Performance Statistics** page to view detailed performance information for a job or thread such as: Job name, CPU statistics, Disk I/O rates and counts and Page fault rates. Elapsed performance statistics are calculated over the time interval identified in the Elapsed time field. Use the Refresh Now or Timed Refresh buttons to update the data. You can find more detailed help on the following elements of this window:

- **Job**

Name of the job under which the thread is running.

- **User**  
User name of the job under which the thread is running.
- **Number**  
System-assigned number of the job under which the thread is running.
- **Thread**  
Identifies the specific thread within the job. Note: The Thread field only appears when the elapsed performance statistics are being shown for a thread.
- **Last refresh**  
The date and time the elapsed performance statistics were last updated.
- **Elapsed time**  
The time interval over which the elapsed performance statistics have been calculated.
- **Performance statistics calculated over elapsed time**
  - **CPU**  
The following performance statistics related to the central processing unit usage are displayed:
    - **Percentage**  
The percentage of available processing unit time that is used by the job or thread during the elapsed time. For multi-processor systems, this value is the average across all processors.
    - **Time**  
The amount of processing unit time that is used by the job or thread during the elapsed time, in milliseconds.
    - **Database percentage**  
The percentage of total processing unit that is used for database processing during the elapsed time. For multi-processor systems, this value is the average across all processors.
    - **Database time**  
The amount of processing unit time that is used for database processing during the elapsed time, in milliseconds.
  - **Disk I/O rate**  
The average number of disk input/output operations performed by the job or thread during the elapsed time, per second. This value is the sum of the asynchronous and synchronous disk input/output operations.
    - **Synchronous**  
The average number, per second, of synchronous(physical) disk I/O operations performed by a job or thread during the elapsed time. This value is the sum of the synchronous database and nondatabase reads and writes.
    - **Asynchronous**  
The average number of asynchronous disk input/output operations performed by the job or thread during the elapsed time, per second. This value is the sum of the asynchronous database and non-database reads and writes.
  - **Disk I/O count**  
The number of disk input/output operations performed by the job or thread during the elapsed time. This value is the sum of the asynchronous and synchronous disk input/output operations.
    - **Synchronous**  
The number of synchronous (physical) disk I/O operations performed by the job or thread during the elapsed time. This value is the sum of the synchronous database and nondatabase reads and writes.
    - **Asynchronous**

The number of asynchronous (physical) disk I/O operations performed by the job or thread during the elapsed time. This value is the sum of the asynchronous database and nondatabase reads and writes.

– **Page fault rate**

The average number of times, per second, that an active program references an address that is not in main storage during the elapsed time.

– **Average response time**

The average interactive transaction response time for the job during the elapsed time, in seconds.

– **Interactive transactions**

The number of user interactions for the job during the elapsed time. For example, the number of times a person pressed a function key. Note: The Average response time field and the Interactive transactions field will not display when the elapsed performance statistics are shown for a thread.

• **Refresh Now**

Click this to refresh the elapsed performance statistics and extend the time period that the statistics are calculated.

• **Timed Refresh...**

Displays a dialog allowing the user to set up automatic refresh of the elapsed performance statistics.

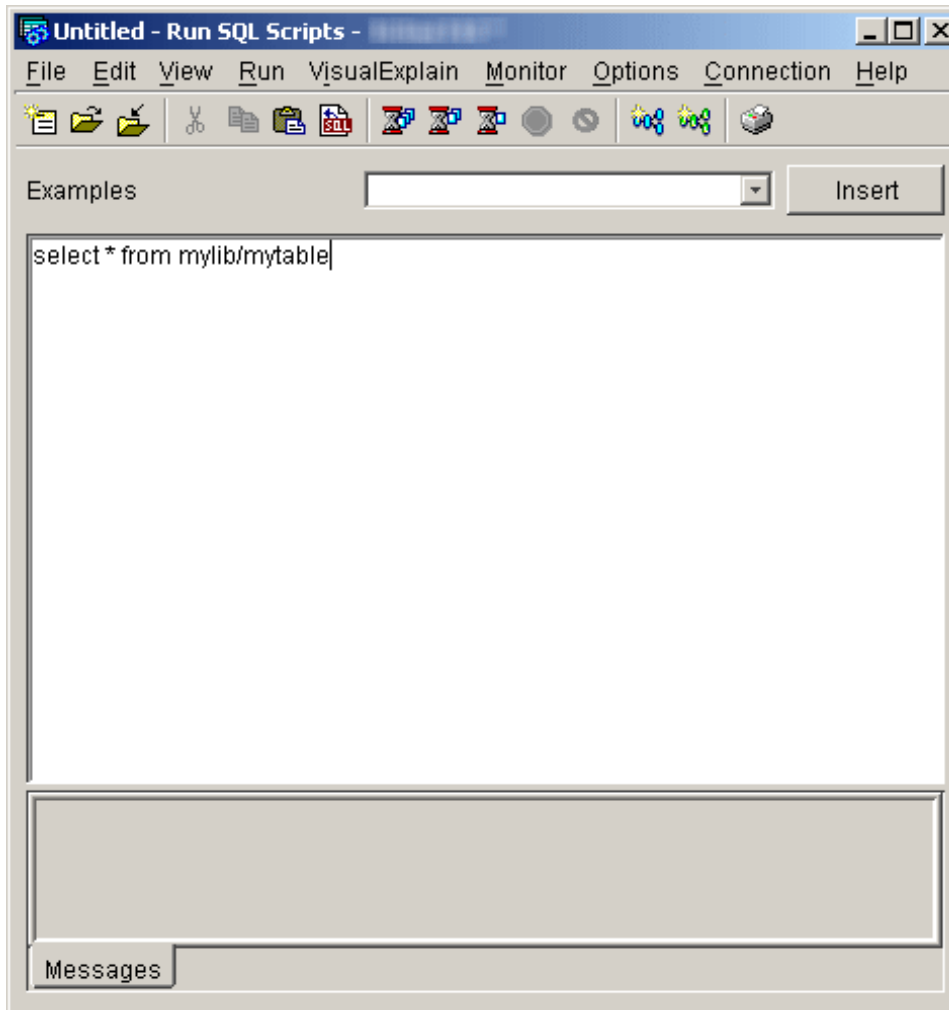
• **Reset Statistics**

Clears the elapsed performance statistics and resets the time period over which the statistics are calculated. Resetting the elapsed performance statistics has no impact on the threads container. Only the current dialog is reset.

### **Last SQL Statement**

Select **Details** then **Last SQL Statement**. The run SQL script window opens if the job has run any SQL statements.

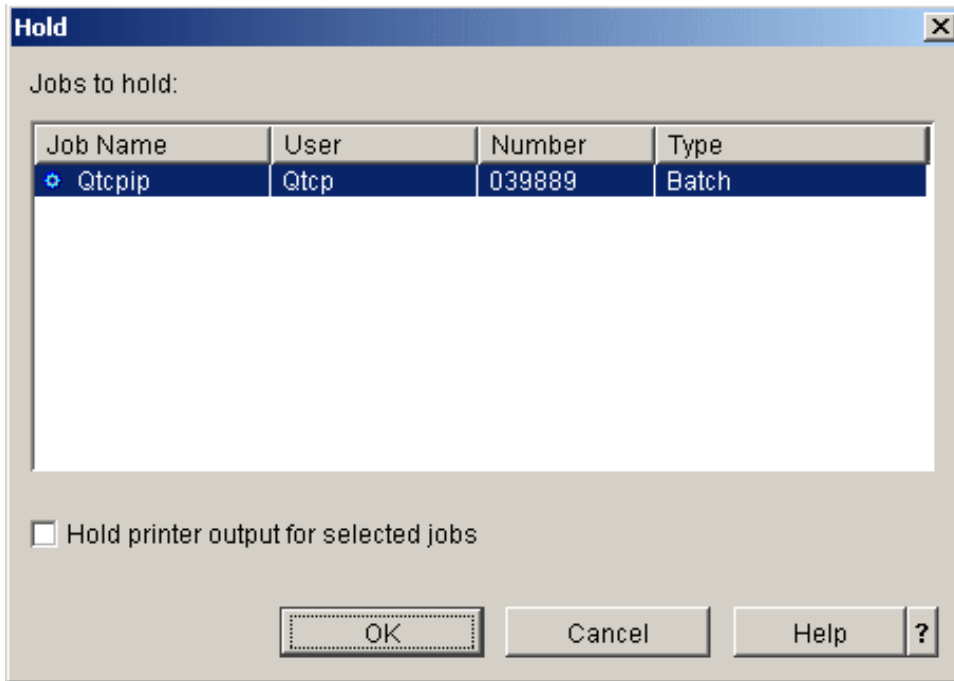




Use the Run SQL Scripts window to create, edit, run, and troubleshoot scripts of SQL statements. When you have finished working with the scripts, you can save them to your PC. See the Run SQL Scripts help topic from the iSeries Navigator for more information.

### Hold

Select **Hold**. The Hold Jobs dialog opens in a new window.



Use the **Hold Jobs** dialog to hold the selected jobs. Holding a job suspends it and prevents it from doing any more work. You can find more detailed help on the following elements of this window:

- **Jobs to hold**

The jobs you selected to hold are displayed. If the selections are not correct, click Cancel or change the selections by deselecting any items displayed. Hold the Ctrl key down and click on the left mouse button to deselect an item in the list.

- **Hold printer output for selected jobs**

Specifies whether the printer output that is created by the job being held is also held. Holding printer output means that the files are held on the output queue and are not printed. Possible options are:

- **Yes**

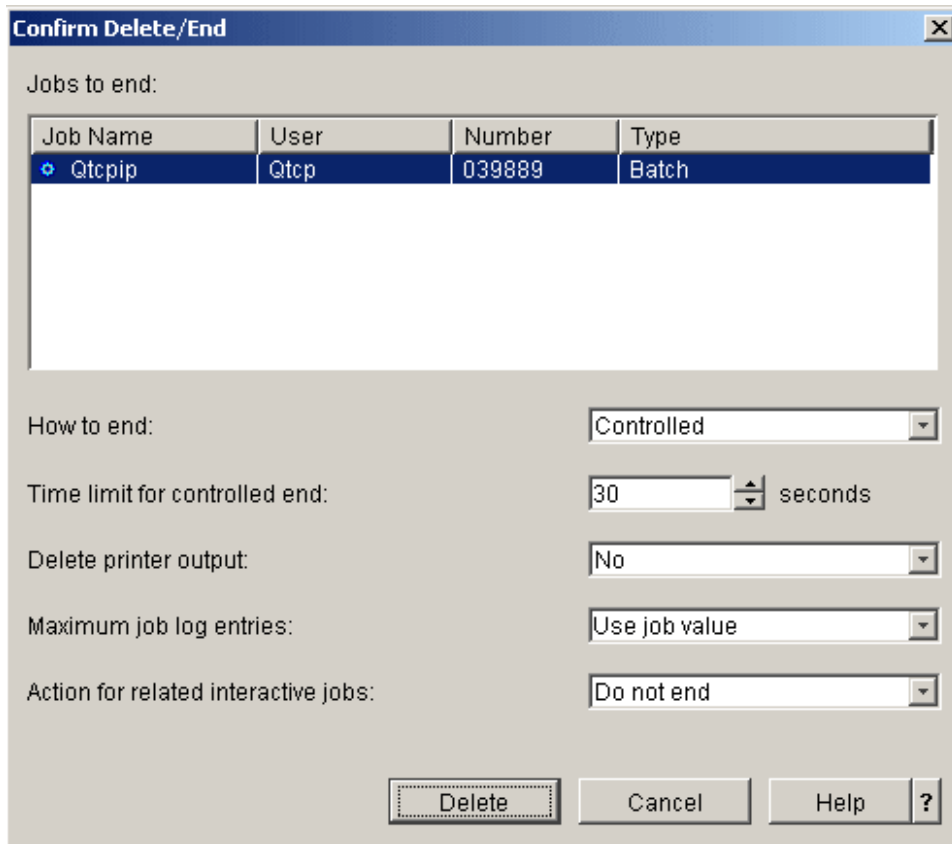
The printer output for the job being held is also held.

- **No**

The printer output for the job being held is not held and is available for printing.

## Delete/End

Select **Delete/End**. The Confirm Delete/End dialog opens in a new window.



Use the **Confirm Delete/End** dialog to delete selected jobs. To delete a job means the job's processing is ended. The jobs are ended when you click **Delete**. You can find more detailed help on the following elements of this window:

- **Jobs to end**

The jobs you selected to end or delete are displayed. If the selections are not correct, click **Cancel** or change the selections by deselecting any items displayed. Hold down the **Ctrl** key and click on the left mouse button to deselect an item in the list.

- **How to end**

Specifies how you want the job to end. Possible options are:

- **Controlled**

The job is allowed to end in a controlled manner within the time limit you specify. The job allows the application program to perform end-of-job processing.

- **Immediately**

The job is ended immediately with no delay time. The application program does not get a chance to perform end-of-job processing as it does when a **Controlled** end is specified. Note: This option may cause undesirable results if data has been partially updated. Therefore, this option should be used only after a controlled end has been attempted unsuccessfully.

- **Time limit for controlled end**

Specify the amount of time the application program has to perform end-of-job processing. Possible values are 1-999999 seconds.

- **Delete printer output**

Select whether printer output created by the job and not yet printed should also be deleted. Possible options are:

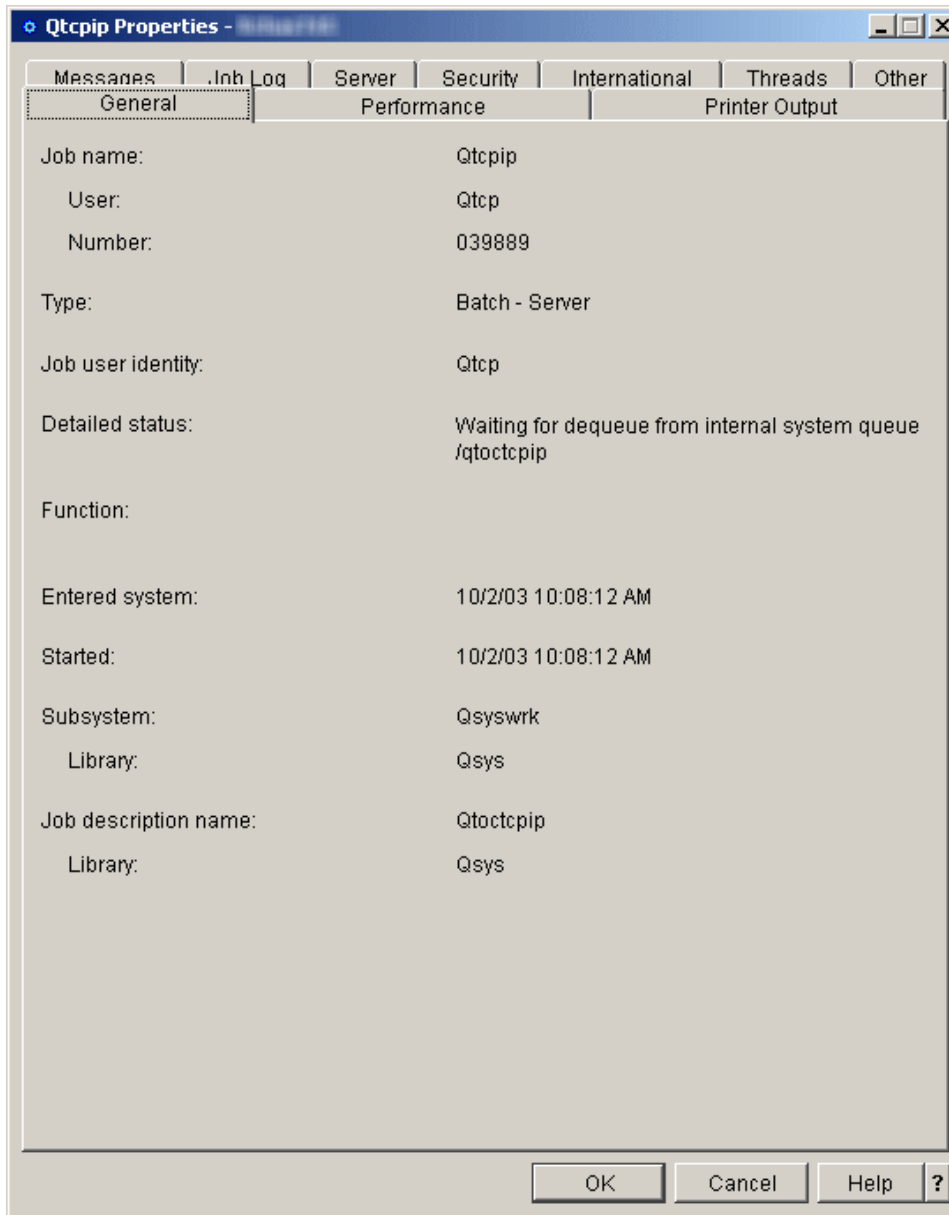
- **Yes**

The printer output for the job being ended is deleted and not printed.

- **No**  
The printer output for the job being ended is kept and is available for printing.
- **Maximum job log entries**  
Select the maximum number of entries you want written to the job log of the job being ended. Possible options are:
  - **Number**  
Specify the maximum number of messages that are written to the job log. This value is the maximum only if it is entered before the job log contains that many messages. Otherwise, the limit stops only the process of writing any more messages to the job log. If 0 is specified before any messages are written to the job log, no job log is produced. Possible values are 0-2147483647.
  - **Use job value**  
The maximum job log entries value does not change. All jobs are started with a value of No maximum, which will be the current job value unless it was subsequently changed, for example, by a previous delete.
  - **No maximum**  
There is no limit to the number of messages that are written to the job log.
- **Action for related interactive jobs**  
Specify how related additional interactive jobs are handled when the selected jobs are ended. Possible choices are:
  - **Do not end**  
Only the selected jobs are ended.
  - **End for group jobs**  
If the selected job is a group job, all jobs for the group are ended. If the job is not a group job, the selected job is ended.
  - **End all**  
All interactive jobs running on the workstation for the selected job are ended, including group jobs and secondary jobs.

## Properties

Select **Properties**. The Job Properties dialog opens in a new window.



For more information on the job properties dialog, see job properties in the iSeries Information Center.

## Interactive subsystem configuration

Interactive users are users who run 5250 display station sessions. These sessions are connected from twinax, remote workstation, Telnet, 5250 Display Station Pass-through, Virtual Terminal API-based applications, and various web-based connections, such as WebFacing, IBM<sup>(R)</sup> WebSphere<sup>(R)</sup> Host Access Transformation Server (HATS), and iSeries<sup>(TM)</sup> Access for the Web 5250 support.

The following steps tell you how to set up a new interactive subsystem. It is recommended that the number of devices allocated to a single interactive subsystem be limited to 250-300, so you should create the appropriate number of interactive subsystems given the number of users on your system.

One of the first planning steps is to determine the naming convention you will use for your subsystems. The naming convention could be something as simple as INTER1, INTER2, INTER3, etc., something more

reflective of the type of work (INVENTORY, ORDERENT, PGMR) or reflective of the geographic location in which your users reside (EAST, CENTRAL, WEST). Decide upon a naming convention that will be meaningful for your system administration.

The steps below are described as if the commands are entered manually. However, you should use a CL program to create your subsystems so you can easily recreate your configurations for recovery purposes.

1. Create a library to store your subsystem configuration objects in. In this example, we use SBSLIB.

```
CRTLIB SBSLIB TEXT('Library to hold subsystem configuration objects')
```

2. Create a class. The class defines certain performance characteristics for your interactive subsystem. 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 a class for each interactive subsystem. Which you choose depends upon whether you want to customize some of the performance settings for particular subsystems. 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. So if you do NOT create a class for each subsystem with the same name as the subsystem, you will need to specify the class name on the Add Routing Entry (ADDRTGE) command since the default for the CLS parameter is \*SBSD, meaning the class name has the same name as the subsystem description.

For example, you may want to have your users doing business critical work to have a higher run priority than the remainder of your users. You could accomplish this by having the users doing business critical work run in a separate subsystem configured with a class that specifies a higher run priority, for example RUNPTY(15) (remember for run priority that a lower number gives a higher priority).

3. Create the subsystem description. Repeat this step for each subsystem you need to define.

```
CRTSBSD SBSDB(SBSLIB/INTER1) POOLS((1 *BASE) (2 *INTERACT)) SGNDSPF(*QDSIGNON)
```

This creates a subsystem description with attributes identical to those of QINTER.

The storage pools used are specified on the subsystem description. If you want to isolate a set of users to a pool of their own, you can do this by specifying a pool ID with a storage size and activity level specifically for the subsystem, rather than using the shared \*INTERACT pool. Optionally, you could define a shared pool with the Change Shared Pool (CHGSHRPOOL) command and specify that shared pool on the subsystem description.

The subsystem description is also where you define the number of jobs you want to run in the subsystem. There are several different options to consider:

- Activity Level on the POOLS parameter on the CRTSBSD command. The activity level determines the maximum number of threads that can be actively running in a pool at one time. More threads than this can be active within the pool, but the activity level determines the number that can be actively running at any given point in time.
- Maximum Jobs (MAXJOBS) parameter on the CRTSBSD command. The maximum number of jobs determines the number of user jobs running in the subsystem. This value cannot be exceeded. Any attempt to start additional jobs when the maximum number of jobs is already running will fail.
- Maximum active routing steps (MAXACT) parameter on the Add Routing Entry (ADDRTGE) command. This is the maximum number of jobs that can be active through this routing entry. It is recommended to use the default value of \*NOMAX.
- Maximum active jobs (MAXACT) parameter on the Add Workstation Entry (ADDWSE) command. This is the maximum number of active jobs that can be active at the same time through the workstation entry. It is recommended to use the default value of \*NOMAX.
- Maximum active jobs (MAXACT) parameter on the Add Job Queue Entry (ADDJOBQE) command. This is the maximum number of jobs that can be active at the same time from the job queue. Note that the default for this parameter is one, so be sure to configure this value appropriately.

The Maximum Active (MAXACT) parameters on the routing entry and workstation entry are available, but add complexity and may make it easier to unintentionally prevent users from accessing the system, thus the recommendation to not use these parameters.

You can also have a custom sign-on display file for each subsystem. The additional parameter SGNDSPF(\*QDSIGNON) specifies that the subsystem use the system supplied sign-on display file, QDSIGNON. You can create your own custom sign-on display file and specify the name here when the subsystem is created. If you want to create your own sign-on display file, see the Work Management book for details.

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. This step is required if you need to use the Transfer Job (TFRJOB) command to transfer jobs into your custom subsystems.

However, by adding a job queue entry to an interactive subsystem, you have also provided the ability to submit batch work to your interactive subsystem. If you need to ensure that no batch work can run in your interactive subsystem, you should not create the job queue and should not add the job queue entry. Without the job queue entry you will not be able to use the Transfer Job (TFRJOB) command, so you must define workstation entries (discussed later in this article) to get users to the correct subsystem.

```
CRTJOBQ JOBQ(SBSLIB/INTER1)
ADDJOBQE SBSL(SBSLIB/INTER1) JOBQ(SBSLIB/INTER1) MAXACT(*NOMAX)
```

5. Add a routing entry to the subsystem. Add the following routing entry. If you look at the routing entries shipped on the system for QINTER, you will see there are some additional routing entries shipped. If you need those functions, add those routing entries to your customized subsystem descriptions as well.

```
ADDRTGE SBSL(SBSLIB/INTER1) SEQNBR(9999) CMPVAL(*ANY) PGM(QSYS/QCMD) POOLID(2)
```

**Note:** The ADDRTGE command specifies which pool identifier (POOLID) to use, the default on the command is 1. In this example, we will specify 2, which is the \*INTERACT pool. If you had set up your subsystem description with dedicated pools, be sure to specify the appropriate pool identifier on the routing entry.

Also, if the name of your class is different from the name of your subsystem description, you will need to specify the class on the ADDRTGE command.

You can get quite complex with routing entries, setting them up so that users can be directly taken to the application when they sign on. In addition, multiple routing entries can be used to set up different performance characteristics for multiple users within a subsystem. If you need to investigate the options available by using routing entries, please refer to the Work Management book.

6. Add workstation entries to the subsystem description. This is the key step for assigning which devices are allocated to which subsystem.

The devices an interactive subsystem allocates is determined by the workstation entries added to the subsystem description. The workstation entry identifies either the workstation name or type, and the allocation it should do for those workstations. The key to this is the Allocation parameter (AT).

AT(\*SIGNON) tells the subsystem that it should attempt to allocate these devices and put up a sign-on display when possible. AT(\*ENTER) tells the subsystem not to allocate the device and not to put up a sign-on display. However, the AT(\*ENTER) does allow the specified workstation devices to transfer into that subsystem with the transfer job (TFRJOB) command.

You need to determine which subsystems should allocate which devices. In addition, determine if you need to allow the use of TFRJOB from one subsystem to another.

A few different examples follow to demonstrate techniques for splitting up the work. These examples focus on identifying which devices the subsystems will attempt to allocate (the next step after this will demonstrate how to set up the workstation entries for those devices that a subsystem should not allocate).

- a. The first example assumes that the primary method to access the system is either using Telnet or Pass-through, and the system default device naming convention is used. In this scenario, all devices use the QPADEVxxxx naming convention. This example simply allocates devices to subsystems based upon the device names.

In the QPADEVxxxx naming convention, numbers (0-9) and letters (consonants B-Z, but not vowels A, E, I, O, U, Y) are used for the last four characters of the device name, so 0001-00B0 gives 300 different device names. Assume there are 900 users to split up among three subsystems, and in this scenario, there is no preference about which subsystem the users run in.

Since all automatically created devices are named QPADEVxxxx, one simple way to split up the work is to have QPADEV0001 through QPADEV00B0 go to the first subsystem, QPADEV00B1 through QPADEV00M0 go to the second subsystem, and QPADEV00M1 - QPADEV00Z0 go to the third subsystem. This simple approach will work, but the system always selects the devices starting with QPADEV0001 and upward. This means the first 300 users go to subsystem INTER1, the next 300 go to INTER2, and the final 300 go to INTER3. However, that doesn't equally distribute the work over the subsystems for scenarios when many users all attempt to sign on at one time.

To better spread the workload across the multiple subsystems, the devices could be assigned to subsystems as shown in the example below. While this approach provides better distribution of work across multiple subsystems, it makes the definition of the workstation entries more complex.

- INTER1 would allocate:

```
ADDWSE SBSDB(SBSLIB/INTER1) WRKSTN(QPADEV000*) AT(*SIGNON)
ADDWSE SBSDB(SBSLIB/INTER1) WRKSTN(QPADEV003*) AT(*SIGNON)
ADDWSE SBSDB(SBSLIB/INTER1) WRKSTN(QPADEV006*) AT(*SIGNON)
ADDWSE SBSDB(SBSLIB/INTER1) WRKSTN(QPADEV009*) AT(*SIGNON)
ADDWSE SBSDB(SBSLIB/INTER1) WRKSTN(QPADEV00D*) AT(*SIGNON)
ADDWSE SBSDB(SBSLIB/INTER1) WRKSTN(QPADEV00H*) AT(*SIGNON)
ADDWSE SBSDB(SBSLIB/INTER1) WRKSTN(QPADEV00L*) AT(*SIGNON)
ADDWSE SBSDB(SBSLIB/INTER1) WRKSTN(QPADEV00P*) AT(*SIGNON)
ADDWSE SBSDB(SBSLIB/INTER1) WRKSTN(QPADEV00S*) AT(*SIGNON)
ADDWSE SBSDB(SBSLIB/INTER1) WRKSTN(QPADEV00W*) AT(*SIGNON)
```

- INTER2 would allocate:

```
ADDWSE SBSDB(SBSLIB/INTER2) WRKSTN(QPADEV001*) AT(*SIGNON)
ADDWSE SBSDB(SBSLIB/INTER2) WRKSTN(QPADEV004*) AT(*SIGNON)
ADDWSE SBSDB(SBSLIB/INTER2) WRKSTN(QPADEV007*) AT(*SIGNON)
ADDWSE SBSDB(SBSLIB/INTER2) WRKSTN(QPADEV00B*) AT(*SIGNON)
ADDWSE SBSDB(SBSLIB/INTER2) WRKSTN(QPADEV00F*) AT(*SIGNON)
ADDWSE SBSDB(SBSLIB/INTER2) WRKSTN(QPADEV00J*) AT(*SIGNON)
ADDWSE SBSDB(SBSLIB/INTER2) WRKSTN(QPADEV00M*) AT(*SIGNON)
ADDWSE SBSDB(SBSLIB/INTER2) WRKSTN(QPADEV00Q*) AT(*SIGNON)
ADDWSE SBSDB(SBSLIB/INTER2) WRKSTN(QPADEV00T*) AT(*SIGNON)
ADDWSE SBSDB(SBSLIB/INTER2) WRKSTN(QPADEV00X*) AT(*SIGNON)
```

- INTER3 would allocate:

```
ADDWSE SBSDB(SBSLIB/INTER3) WRKSTN(QPADEV002*) AT(*SIGNON)
ADDWSE SBSDB(SBSLIB/INTER3) WRKSTN(QPADEV005*) AT(*SIGNON)
ADDWSE SBSDB(SBSLIB/INTER3) WRKSTN(QPADEV008*) AT(*SIGNON)
ADDWSE SBSDB(SBSLIB/INTER3) WRKSTN(QPADEV00C*) AT(*SIGNON)
ADDWSE SBSDB(SBSLIB/INTER3) WRKSTN(QPADEV00G*) AT(*SIGNON)
ADDWSE SBSDB(SBSLIB/INTER3) WRKSTN(QPADEV00K*) AT(*SIGNON)
ADDWSE SBSDB(SBSLIB/INTER3) WRKSTN(QPADEV00N*) AT(*SIGNON)
ADDWSE SBSDB(SBSLIB/INTER3) WRKSTN(QPADEV00R*) AT(*SIGNON)
ADDWSE SBSDB(SBSLIB/INTER3) WRKSTN(QPADEV00V*) AT(*SIGNON)
ADDWSE SBSDB(SBSLIB/INTER3) WRKSTN(QPADEV00Z*) AT(*SIGNON)
```

- b. The second example assumes that the business is geographically disparate and that a subsystem and device naming convention has been put into place such that users from different locations use different device names and run in different subsystems defined for each geography. For example, east coast users all use devices whose names begin with EAST and run in subsystem EAST. West coast users all use devices whose names begin with WEST and run in subsystem WEST. Likewise, central users all use devices whose names begin with CENTRAL and run in subsystem CENTRAL. In this scenario, each subsystem allocates devices for its geographical area.



```
ADDWSE SBSDB(SBSLIB/EAST) WRKSTN(EAST*) AT(*SIGNON)
ADDWSE SBSDB(SBSLIB/CENTRAL) WRKSTN(CENTRAL*) AT(*SIGNON)
ADDWSE SBSDB(SBSLIB/WEST) WRKSTN(WEST*) AT(*SIGNON)
```

- c. The third example assumes that 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 PGMR 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 of QINTER.

```
ADDWSE SBSDB(SBSLIB/PGMR) WRKSTN(PGMR*) AT(*SIGNON)
ADDWSE SBSDB(SBSLIB/ORDERENT) WRKSTN(ORDERENT*) AT(*SIGNON)
ADDWSE SBSDB(QGPL/QINTER) WRKSTN(QPADEV*) AT(*SIGNON)
```

7. When you begin using your own set of subsystems, you may no longer 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 you want to run under your other subsystems. There are two possible ways to do this.

- a. Remove the \*ALL workstation entry from QINTER, then add specific workstation entries that indicate which devices you want QINTER to allocate.

Remove the workstation type entry of \*ALL is to prevent QINTER from attempting to allocate all workstations.

```
RMVWSE SBSDB(QGPL/QINTER) WRKSTNTYPE(*ALL)
```

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.

```
ADDWSE SBSDB(SBSLIB/QINTER) WRKSTN(DSP*)
```

This is the best solution because it avoids the use of workstation entries with the AT(\*ENTER) keyword. AT(\*ENTER) tells the subsystem not to put up a sign-on display on the device, however a user on that device can use the Transfer Job (TFRJOB) commands to get their job running in that subsystem. If you have no need for TFRJOB, the best solution is to not use workstation entries with AT(\*ENTER). In addition, using workstation entries with AT(\*ENTER) can add system overhead for switched lines devices that use switched disconnect.

- b. Add a workstation entry to tell QINTER not to allocate the devices that are assigned to other subsystems; however, QINTER will 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.

Step 6 above added workstation name entries to the customized subsystems to demonstrate various examples of how to subdivide the work. There were three examples. Below are the workstation name entries that need to be added to QINTER to allow devices other than those allocated to the specific subsystems to be allocated by QINTER.

- 1) For the first example, the following workstation entry informs QINTER that it should not allocate any of the QPADEVxxxx devices

```
ADDWSE SBSDB(QGPL/QINTER) WRKSTN(QPADEV*) AT(*ENTER)
```

- 2) Likewise, for the second example identified above, you would use the following commands:

```
ADDWSE SBSDB(QGPL/QINTER) WRKSTN(EAST) AT(*ENTER)
ADDWSE SBSDB(QGPL/QINTER) WRKSTN(WEST) AT(*ENTER)
ADDWSE SBSDB(QGPL/QINTER) WRKSTN(CENTRAL) AT(*ENTER)
```

- 3) And for the third example:

```
ADDWSE SBSDB(QGPL/QINTER) WRKSTN(PGMR) AT(*ENTER)
ADDWSE SBSDB(QGPL/QINTER) WRKSTN(ORDERENT) AT(*ENTER)
```

8. A final, but VERY important consideration regarding QINTER is the workstation type entry of \*CONS for the console. Be sure you do not accidentally prevent someone from signing on at the console.

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 a good practice always to run the console in the controlling subsystem and 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 into 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, probably not what the operator intended. You prevent this from happening by not adding any workstation entries for the console to your custom interactive subsystems.

“Scenario details: CL program example” has been provided as a starting point for configuring your own interactive subsystems.

Once the subsystems descriptions have been created, use the Display Subsystem Description (DSPSBSD) command to display the various attributes of the subsystem and verify that the set up has been done correctly.

Start the subsystems using the Start Subsystem (STRSBS) command. Sign-on to the system using devices with various names and verify that the job starts in the proper subsystem. You can do this verification with the Work with Subsystem Jobs (WRKSBSJOB) command.

It is a good idea to modify the system startup program to automatically start these subsystems in order to avoid having to manually start them each time the system is taken to the restricted state or restarted. See Changing the IPL Start-up Program for more information on how to change your IPL start-up program.

## Scenario details: CL program example

The following sample source code can be used as a starting point for setting up your customized subsystems.

**Note:** Read the code example disclaimer for important legal information.

Put the source into QGPL/QCLSRC and execute the following commands to create the command and CPP.

```
CRTCMD CMD(QGPL/CRTMLTSBS) PGM(QGPL/QCRTMLTSBS) SRCFILE(QGPL/QCLSRC)
CRTCLPGM PGM(QGPL/QCRTMLTSBS) SRCFILE(QGPL/QCLSRC) LOG(*NO)
```

To run the program, type the command CRTMLTSBS and press the F4 (prompt) key. You will see the following prompt:

```
-----
                          Create multiple subsystems (CRTMLTSBS)

Type choices, press Enter.

Subsystem name prefix . . . . . QINTER____ QINTER, name
Library name . . . . . *CURLIB____ *CURLIB, name
Number of subsystems . . . . . 1____ 1-999
Devices per subsystem . . . . . 200 1-500
-----
```

What this program does:

- Creates up to 999 subsystems using a naming convention of aaaaaaNNN.
  - aaaaaaa is up to a 7 character user supplied name.
  - NNN is 001 to 999 sequential number of the subsystem that is created.
  - The program numbers the subsystems sequentially as they are created.

- Device name entries are added to the subsystems based on the QPADEV.... naming conventions. The program adds the requested number of device names to each subsystem as the subsystem is created. The device names are added “round robin” style to the subsystems. That means that subsystem 001 will have QPADEV0001, subsystem 002 will have QPADEV0002, etc.
- The library specified on the CRTMLTSBS command must exist prior to running the command.
- If the subsystem description exists when the command is executed, that subsystem will not be altered in any way although the device name array will continue to grow. That is, if a subsystem exists and the request was to add 200 devices to each subsystem, those 200 device names, (in alphabetical sequence) will not be used again. Under these circumstances, the program will run to completion and a CPF2227 message will be sent indicating “One or more errors occurred during processing of command” Check the joblog for the actual cause.
- Each subsystem has it’s own jobq entry, jobq name and class.
- Each subsystem has the same basic configuration as the IBM-supplied QINTER subsystem description:

```
-----
Subsystem description . . . . . : QINTER001
Library . . . . . : SBSLIB
Maximum jobs in subsystem . . . . . : *NOMAX
Sign-on display file . . . . . : QDSIGNON
Library . . . . . : QSYS
System library list entry . . . . . : *NONE
-----
```

```
-----
Pool      Storage      Activity
ID        Size (K)      Level
1         *BASE
2         *INTERACT
-----
```

Type options, press Enter.  
5=Display work station name details

```
-----
Opt Name      Opt Name      Opt Name      Opt Name
QPADEV0000
QPADEV0001
QPADEV0002    *** The number of devices will depend on how many were
QPADEV0003      requested.
QPADEV0004
QPADEV....
-----
```

```
-----
Seq Job      Max -----Max by Priority-----
Nbr Queue    Library  Active  1  2  3  4  5  6  7  8  9
10  QINTER001  SBSLIB   *NOMAX  *  *  *  *  *  *  *  *  *
```

Display Routing Entries

System: TESTSYS

Subsystem description: QINTER001      Status: INACTIVE

Type options, press Enter.

5=Display details

Opt	Seq Nbr	Program	Library	Compare Value	Start Pos
	10	QCMD	QSYS	'QCMDI'	1
	20	QCMD	QSYS	'QS36MRT'	1
	40	QARDRIVE	QSYS	'525XTEST'	1
	700	QCL	QSYS	'QCMD38'	1

```
-----
Routing entry sequence number . . . . . : 10
-----
```

```

Program . . . . . : QCMD
  Library . . . . . : QSYS
Class . . . . . : QINTER001
  Library . . . . . : SBSLIB
Maximum active routing steps . . . . . : *NOMAX
Pool identifier . . . . . : 2
Compare value . . . . . : 'QCMDI'

Compare start position . . . . . : 1

```

```

-----
Routing entry sequence number . . . . . : 20
Program . . . . . : QCMD
  Library . . . . . : QSYS
Class . . . . . : QINTER001
  Library . . . . . : SBSLIB
Maximum active routing steps . . . . . : *NOMAX
Pool identifier . . . . . : 2
Compare value . . . . . : 'QS36MRT'

Compare start position . . . . . : 1

```

```

-----
Routing entry sequence number . . . . . : 40
Program . . . . . : QARDRIVE
  Library . . . . . : QSYS
Class . . . . . : QINTER001
  Library . . . . . : SBSLIB
Maximum active routing steps . . . . . : *NOMAX
Pool identifier . . . . . : 2
Compare value . . . . . : '525XTEST'

Compare start position . . . . . : 1

```

```

-----
Routing entry sequence number . . . . . : 700
Program . . . . . : QCL
  Library . . . . . : QSYS
Class . . . . . : QINTER001
  Library . . . . . : SBSLIB
Maximum active routing steps . . . . . : *NOMAX
Pool identifier . . . . . : 2
Compare value . . . . . : 'QCMD38'

Compare start position . . . . . : 1

```

```

-----
Routing entry sequence number . . . . . : 9999
Program . . . . . : QCMD
  Library . . . . . : QSYS
Class . . . . . : QINTER001
  Library . . . . . : SBSLIB
Maximum active routing steps . . . . . : *NOMAX
Pool identifier . . . . . : 2
Compare value . . . . . : *ANY

Compare start position . . . . . :

```

**The source code for the command follows:**

```

/*****/
/* 5722-SS1 (C) COPYRIGHT IBM CORP. 2003, 2003 */
/* LICENSED MATERIAL - PROGRAM PROPERTY OF IBM */
/* */
/*****/
/* Create this command using the following command: */
/* */
/* Assuming the source is in file QGPL/QCLSRC */
/* */
/* CRTCMD CMD(QGPL/CRTMLTSBS) PGM(QGPL/QCRTMLTSBS) SRCFILE(QGPL/QCLSRC) */
/* */
/*****/
/* */
      CMD          PROMPT('Create multiple subsystems')
      PARM         KWD(SBSNAM) TYPE(*CHAR) LEN(7) DFT(QINTER) +
                  CHOICE('QINTER, name') PROMPT('Subsystem +
                  name prefix')
      PARM         KWD(SBSLIB) TYPE(*CHAR) LEN(10) DFT(*CURLIB) +
                  CHOICE('*CURLIB, name') PROMPT('Library +
                  name')
      PARM         KWD(NBRSBS) TYPE(*DEC) LEN(3) DFT(1) RANGE(1 +
                  999) CHOICE(*VALUES) PROMPT('Number of +
                  subsystems')
      PARM         KWD(NBRDEV) TYPE(*DEC) LEN(3) DFT(200) +
                  RANGE(1 300) CHOICE(*VALUES) +
                  PROMPT('Devices per subsystem')
/* */

```

**The source code for the CL program follows:**

```

/*****/
/* Assuming the source member is in QGPL/QCLSRC file. */
/* Create this program with the following command: */
/* */
/* CRTCLPGM PGM(QGPL/QCRTMLTSBS) SRCFILE(QGPL/QCLSRC) LOG(*NO) */
/* */
/* This program must be invoked using the CRTMLTSBS command. */
/* */
/*****/
/* */
      PGM          PARM(&SBSNAM &SBSLIB &NBRSBS &NBRDEV)
/* */
/*****/
/* 5722-SS1 (C) COPYRIGHT IBM CORP. 2003, 2003 */
/* LICENSED MATERIAL - PROGRAM PROPERTY OF IBM */
/*****/
/* */
      QSYS/DCL    VAR(&CPYR) TYPE(*CHAR) LEN(230) +
                  VALUE('LICENSED MATERIALS PROPERTY OF IBM +
                  5722SS1 (C) COPYRIGHT IBM CORP. 2003, +
                  2003 ALL RIGHTS RESERVED. US GOVERNMENT +
                  USERS RESTRICTED RIGHTS - USE, DUPLICATION +
                  OR DISCLOSURE RESTRICTED BY GSA ADP +
                  SCHEDULE CONTRACT WITH IBM CORP.')
```

```

/* */
/*****/
/* */
      DCL         VAR(&SBSNAM) TYPE(*CHAR) LEN(7)
      DCL         VAR(&SBSLIB) TYPE(*CHAR) LEN(10)
      DCL         VAR(&NBRSBS) TYPE(*DEC) LEN(3 0)
      DCL         VAR(&NBRDEV) TYPE(*DEC) LEN(3 0)
      DCL         VAR(&SBSCNT) TYPE(*DEC) LEN(3 0)
      DCL         VAR(&SBSCNTC) TYPE(*CHAR) LEN(3)
      DCL         VAR(&DEVcnt) TYPE(*DEC) LEN(10 0)
      DCL         VAR(&SBSNAME) TYPE(*CHAR) LEN(10)
      DCL         VAR(&DEVNAME) TYPE(*CHAR) LEN(10)
      DCL         VAR(&POS1) TYPE(*DEC) LEN(2 0) VALUE(1)

```

```

DCL      VAR(&POS2) TYPE(*DEC) LEN(2 0) VALUE(1)
DCL      VAR(&POS3) TYPE(*DEC) LEN(2 0) VALUE(1)
DCL      VAR(&POS4) TYPE(*DEC) LEN(2 0) VALUE(1)
DCL      VAR(&TMP4) TYPE(*CHAR) LEN(4)
DCL      VAR(&MSGID) TYPE(*CHAR) LEN(7)
DCL      VAR(&MSGDTA) TYPE(*CHAR) LEN(512)
DCL      VAR(&KEYVAR) TYPE(*CHAR) LEN(4)
DCL      VAR(&MSGFILE) TYPE(*CHAR) LEN(10)
DCL      VAR(&MSGLIB) TYPE(*CHAR) LEN(10)
DCL      VAR(&ERR) TYPE(*DEC) LEN(1 0) VALUE(0)
DCL      VAR(&ARY) TYPE(*CHAR) LEN(31) +
          VALUE('0123456789BCDFGHJKLMNPQRSTVWXYZ')

/*
/*****
/* Monitor for any unexpected condition then resignal the exception to
/* the caller.
/*****
/*
      MONMSG      MSGID(CPF0000) EXEC(GOTO CMDLBL(ERRORXIT))
/*
/*****
/* Insert the copyright.
/*****
/*
      CHGVAR      VAR(&CPYR) VALUE(&CPYR)
/*
/*****
/* Insert the copyright.
/*****
/*
      CHGVAR      VAR(&DEVCNT) VALUE(&NBRDEV * &SBSCNT)
/*
/*****
/* Set the library name if necessary.
/*****
/*
      IF          COND(&SBSLIB *EQ '*CURLIB') THEN(DO)
RTVJOBA      CURLIB(&SBSLIB)
ENDDO
      IF          COND(&SBSLIB *EQ '*NONE') THEN(DO)
CHGVAR      VAR(&SBSLIB) VALUE('QGPL')
ENDDO
/*
/*****
/* Make sure the specified library exists.
/*****
/*
      CHKOBJ      OBJ(QSYS/&SBSLIB) OBJTYPE(*LIB)
/*
/*****
/* Put out a status message.
/*****
/*
      CHGVAR      VAR(&MSGDTA) VALUE('Checking for existing +
          subsystems')
/*
      SNDPGMMSG  MSGID(CPF9898) MSGF(QSYS/QCPFMSG) +
          MSGDTA(&MSGDTA) TOPGMQ(*EXT) MSGTYPE(*STATUS)
/*
/*****
/* Make sure none of the subsystems exist.
/*****
/*
CHKFORSBS:
      CHGVAR      VAR(&SBSCNT) VALUE(&SBSCNT + 1)
/*
      IF          COND(&SBSCNT *GT &NBR SBS) THEN(DO)

```

```

        GOTO      CMDLBL(NOSBS)
        ENDDO

/*
        CHGVAR   VAR(&SBSCNTC) VALUE(&SBSCNT)
        CHGVAR   VAR(&SBSNAME) VALUE(&SBSNAM *TCAT &SBSCNTC)
/*
/*****
/* Check for the existence of the subsystem.
/*****
/*
        CHKOBJ   OBJ(&SBSLIB/&SBSNAME) OBJTYPE(*SBSD)
        MONMSG   MSGID(CPF9801) EXEC(DO)
        RCVMSG   MSGTYPE(*EXCP) RMV(*YES)
        GOTO     CMDLBL(CHKFORSBS)
        ENDDO

/*
/*****
/* If a subsystem already exists, signal the error and return.
/*****
/*
        SNDPGMMSG MSGID(CPD1411) MSGF(QSYS/QCPFMSG) +
                MSGDTA(&SBSNAME *CAT &SBSLIB) +
                TOPGMQ(*SAME) MSGTYPE(*INFO)
/*
        SNDPGMMSG MSGID(CPF1696) MSGF(QSYS/QCPFMSG) +
                MSGDTA(&SBSNAME) +
                TOPGMQ(*SAME) MSGTYPE(*DIAG)
/*
        CHGVAR   VAR(&ERR) VALUE(1)
/*
        GOTO     CMDLBL(END)
NOSBS:
        CHGVAR   VAR(&SBSCNT) VALUE(0)
/*
/*****
/* Main loop of the program.
/*****
/*
NEXTSBS:
        CHGVAR   VAR(&SBSCNT) VALUE(&SBSCNT + 1)
        CHGVAR   VAR(&SBSCNTC) VALUE(&SBSCNT)
        CHGVAR   VAR(&SBSNAME) VALUE(&SBSNAM *TCAT &SBSCNTC)
/*
/*****
/* Put out a status message.
/*****
/*
        CHGVAR   VAR(&MSGDTA) VALUE('Creating subsystem' +
                *BCAT &SBSNAME)
/*
        SNDPGMMSG MSGID(CPF9898) MSGF(QSYS/QCPFMSG) +
                MSGDTA(&MSGDTA) TOPGMQ(*EXT) MSGTYPE(*STATUS)
/*
/*****
/* If the subsystem doesn't exist, create it and the associated entries.
/*****
/*
/* Create the subsystem description.
/*
/*
        CRTSBSD   SBSDB(&SBSLIB/&SBSNAME) POOLS((1 *BASE) (2 +
                *INTERACT)) TEXT('Created by the +
                CRTMLTSBS command')
        RCVMSG   MSGTYPE(*COMP) RMV(*YES)
/*
/* Create a jobq.
/*

```

```

        CHKOBJ      OBJ(&SBSLIB/&SBSNAME) OBJTYPE(*JOBQ)
        MONMSG      MSGID(CPF9801) EXEC(DO)
        RCVMSG      MSGTYPE(*EXCP) RMV(*YES)
        CRTJOBQ     JOBQ(&SBSLIB/&SBSNAME) TEXT('Created by the +
                   CRTMLTSBS command')
        RCVMSG      MSGTYPE(*COMP) RMV(*YES)
        GOTO        CMDLBL(ADDJOBQE)
        ENDDO

/*                                                    */
        CHGVAR      VAR(&ERR) VALUE(1)
/*                                                    */
        SNDPGMMSG   MSGID(CPF3323) MSGF(QSYS/QCPFMSG) +
                   MSGDTA(&SBSNAME *CAT &SBSLIB) +
                   TOPGMQ(*SAME) MSGTYPE(*DIAG)
/*                                                    */
/* Add the jobq entry.                                */
/*                                                    */
ADDJOBQE:
        ADDJOBQE    SBSD(&SBSLIB/&SBSNAME) +
                   JOBQ(&SBSLIB/&SBSNAME) MAXACT(*NOMAX)
        RCVMSG      MSGTYPE(*COMP) RMV(*YES)
/*                                                    */
/* Create the class                                  */
/*                                                    */
        CHKOBJ      OBJ(&SBSLIB/&SBSNAME) OBJTYPE(*CLS)
        MONMSG      MSGID(CPF9801) EXEC(DO)
        RCVMSG      MSGTYPE(*EXCP) RMV(*YES)
        CRTCLS     CLS(&SBSLIB/&SBSNAME) RUNPTY(20) PURGE(*YES) +
                   DFTWAIT(30) CPUTIME(*NOMAX) +
                   MAXTMPSTG(*NOMAX) MAXTHD(*NOMAX) +
                   TEXT('Created by the CRTMLTSBS command')
        RCVMSG      MSGTYPE(*COMP) RMV(*YES)
        GOTO        CMDLBL(ADDRTGE)
        ENDDO

/*                                                    */
        CHGVAR      VAR(&ERR) VALUE(1)
        SNDPGMMSG   MSGID(CPF1064) MSGF(QSYS/QCPFMSG) +
                   MSGDTA(&SBSNAME *CAT &SBSLIB) +
                   TOPGMQ(*SAME) MSGTYPE(*DIAG)
/*                                                    */
/* Add the standard routing entries.                  */
/*                                                    */
ADDRTGE:
        ADDRTGE     SBSD(&SBSLIB/&SBSNAME) SEQNBR(10) +
                   CMPVAL('QCMDI' 1) PGM(QSYS/QCMD) +
                   CLS(&SBSLIB/&SBSNAME) POOLID(2)
        RCVMSG      MSGTYPE(*COMP) RMV(*YES)
/*                                                    */
        ADDRTGE     SBSD(&SBSLIB/&SBSNAME) SEQNBR(20) +
                   CMPVAL('QS36MRT' 1) PGM(QSYS/QCMD) +
                   CLS(&SBSLIB/&SBSNAME) POOLID(2)
        RCVMSG      MSGTYPE(*COMP) RMV(*YES)
/*                                                    */
        ADDRTGE     SBSD(&SBSLIB/&SBSNAME) SEQNBR(40) +
                   CMPVAL('525XTEST' 1) PGM(QSYS/QARDRIVE) +
                   CLS(&SBSLIB/&SBSNAME) POOLID(2)
        RCVMSG      MSGTYPE(*COMP) RMV(*YES)
/*                                                    */
        ADDRTGE     SBSD(&SBSLIB/&SBSNAME) SEQNBR(700) +
                   CMPVAL('QCMD38' 1) PGM(QSYS/QCL) +
                   CLS(&SBSLIB/&SBSNAME) POOLID(2)
        RCVMSG      MSGTYPE(*COMP) RMV(*YES)
/*                                                    */
        ADDRTGE     SBSD(&SBSLIB/&SBSNAME) SEQNBR(9999) +
                   CMPVAL(*ANY) PGM(QSYS/QCMD) +
                   CLS(&SBSLIB/&SBSNAME) POOLID(2)
        RCVMSG      MSGTYPE(*COMP) RMV(*YES)

```



```

/*                                                                    */
/* Go create the next subsystem.                                       */
/*                                                                    */
      IF          COND(&SBSCNT *LT &NBRSBS) THEN(DO)
      GOTO        CMDLBL(NEXTSBS)
      ENDDO

/*                                                                    */
/*****                                                                    */
/* Put out a status message.                                           */
/*****                                                                    */
/*                                                                    */
      CHGVAR      VAR(&MSGDTA) VALUE('Adding device names to +
      the' *BCAT &SBSNAM *TCAT '... subsystems')

/*                                                                    */
      SNDPGMMSG   MSGID(CPF9898) MSGF(QSYS/QCPFMSG) +
      MSGDTA(&MSGDTA) TOPGMQ(*EXT) MSGTYPE(*STATUS)

/*                                                                    */
/*****                                                                    */
/* Generate a device name.                                             */
/*****                                                                    */
/*                                                                    */
NEXTDEV:
      CHGVAR      VAR(&POS4) VALUE(&POS4 + 1)
      IF          COND(&POS4 *GT 31) THEN(DO)
      CHGVAR      VAR(&POS4) VALUE(1)
      CHGVAR      VAR(&POS3) VALUE(&POS3 + 1)
      ENDDO
      IF          COND(&POS3 *GT 31) THEN(DO)
      CHGVAR      VAR(&POS3) VALUE(1)
      CHGVAR      VAR(&POS2) VALUE(&POS2 + 1)
      ENDDO
      IF          COND(&POS2 *GT 31) THEN(DO)
      CHGVAR      VAR(&POS2) VALUE(1)
      CHGVAR      VAR(&POS1) VALUE(&POS1 + 1)
      ENDDO

/*                                                                    */
/* Check for using up all device names.                                 */
/*                                                                    */
      IF          COND(&POS1 *GT 31) THEN(DO)

/*                                                                    */
/* We ran out of device names.                                         */
/*                                                                    */
      CHGVAR      VAR(&MSGDTA) VALUE('We used all possible +
      device names')
      SNDPGMMSG   MSGID(CPF9898) MSGF(QSYS/QCPFMSG) +
      MSGDTA(&MSGDTA) TOPGMQ(*PRV) MSGTYPE(*ESCAPE)

      RETURN
      ENDDO

/*                                                                    */
      CHGVAR      VAR(%SST(&TMP4 1 1)) VALUE(%SST(&ARY &POS1 1))
      CHGVAR      VAR(%SST(&TMP4 2 1)) VALUE(%SST(&ARY &POS2 1))
      CHGVAR      VAR(%SST(&TMP4 3 1)) VALUE(%SST(&ARY &POS3 1))
      CHGVAR      VAR(%SST(&TMP4 4 1)) VALUE(%SST(&ARY &POS4 1))
      CHGVAR      VAR(&DEVNAME) VALUE('QPADEV' *TCAT &TMP4)

/*                                                                    */
/* Set the subsystem name.                                             */
/*                                                                    */
      IF          COND(&SBSCNT *GE &NBRSBS) THEN(DO)
      CHGVAR      VAR(&SBSCNT) VALUE(0)
      ENDDO
      CHGVAR      VAR(&SBSCNT) VALUE(&SBSCNT + 1)
      CHGVAR      VAR(&SBSCNTC) VALUE(&SBSCNT)
      CHGVAR      VAR(&SBSNAME) VALUE(&SBSNAM *TCAT &SBSCNTC)

/*                                                                    */
/* Add the device name to the subsystem.                               */
/*                                                                    */
      ADDWSE      SBSD(&SBSLIB/&SBSNAME) WRKSTN(&DEVNAME) +

```

```

                                JOB(*USRPRF) MAXACT(*NOMAX) AT(*SIGNON)

RCVMSG      MSGTYPE(*LAST) RMV(*NO) MSGID(&MSGID)
IF          COND(&MSGID *EQ 'CPC1602') THEN(DO)
RCVMSG      MSGTYPE(*LAST) RMV(*YES)
ENDDO
ELSE        CMD(DO)
IF          COND(&MSGID *EQ 'CPF1698') THEN(DO)
RCVMSG      MSGTYPE(*LAST) RMV(*YES)
ENDDO
RCVMSG      MSGTYPE(*LAST) RMV(*NO) MSGID(&MSGID)
IF          COND(&MSGID *EQ 'CPD1431') THEN(DO)
RCVMSG      MSGTYPE(*LAST) RMV(*YES)
ENDDO
ENDDO

/*                                                    */
/*                                                    */
/* Keep a count of the number of devices in a subsystem.  */
/*                                                    */
CHGVAR      VAR(&DEVCNT) VALUE(&DEVCNT + 1)
/*                                                    */
/*                                                    */
/* Have we reached the number of devices in a subsystem?  */
/*                                                    */
IF          COND(&DEVCNT *GE (&NBRDEV * &NBR SBS)) THEN(DO)
GOTO        CMDLBL(END)
ENDDO

/*                                                    */
/* Go create another device name.  */
/*                                                    */
GOTO        CMDLBL(NEXTDEV)
/*                                                    */
/*****
/*                                                    */
END:
/*                                                    */
IF          COND(&ERR *EQ 1) THEN(DO)
SNDPGMMSG  MSGID(CPF2227) MSGF(QSYS/QCPFMSG) +
TOPGMQ(*PRV) MSGTYPE(*COMP)
ENDDO
ELSE        CMD(DO)
CHGVAR      VAR(&MSGDTA) VALUE('CRTMLTSBS command +
completed..')
SNDPGMMSG  MSGID(CPF9898) MSGF(QSYS/QCPFMSG) +
MSGDTA(&MSGDTA) TOPGMQ(*PRV) MSGTYPE(*COMP)
ENDDO
RETURN

/*                                                    */
/*****
/*                                                    */
ERRorexIT:
/*                                                    */
/* Receive an exception message and save the variables.  */
/*                                                    */
RCVMSG      PGMQ(*SAME) MSGTYPE(*EXCP) RMV(*YES) +
KEYVAR(&KEYVAR) MSGDTA(&MSGDTA) +
MSGID(&MSGID) MSGF(&MSGFILE) MSGFLIB(&MSGLIB)
MONMSG      MSGID(CPF0000) CMPDTA(*NONE) EXEC(RETURN)
/*                                                    */
/* Resignal the message.  */
/*                                                    */
SNDPGMMSG  MSGID(&MSGID) MSGF(&MSGLIB/&MSGFILE) +
MSGDTA(&MSGDTA) MSGTYPE(*ESCAPE) +
KEYVAR(&KEYVAR)
MONMSG      MSGID(CPF0000) CMPDTA(*NONE) EXEC(RETURN)
/*                                                    */
/*****

```

```

/*                                     */
      RETURN                           */
/*                                     */
/*****                               */
/*                                     */
      ENDPGM                            */
/*                                     */
/*****                               */

```

---

## Managing interactive users and devices

This section discusses some additional topics, such as assigning device names, inactive device time-out, and device recovery action, that are related to interactive job management within OS/400<sup>(R)</sup>.

### Getting users to a specific subsystem

For interactive sessions, OS/400 assigns the jobs to subsystems by device description. This has worked well, particularly when users connected to the system with twinax-attached terminals, because you can identify which users are associated with which display devices. However, as interactive work moved away from twinax terminals to virtual display devices, the association between a device description to a user profile became significantly more difficult.

There are many times when you may want to be able to route your users to a subsystem based upon their user profile rather than their device description. Unfortunately, none of the subsystem configuration capabilities supplied by OS/400 provide this function. To route the work to a subsystem based upon user profile, the typical technique used is to have all users first come into the system through a single subsystem, often QINTER, and specify in the user's initial program in their user profile the transfer job command to transfer the user's job to the desired subsystem. While this approach typically works, there are some important considerations:

- There is some system overhead when you use the Transfer Job (TFRJOB) command. You are not simply moving your job from one subsystem to another, but are starting a new routing step in the target subsystem and ending the routing step in the current subsystem. (For more information on routing steps, see the Work Management book). It is a much more efficient use of system resources to get the users to the desired subsystem without using TFRJOB.
- If you have all of the devices allocated to one subsystem, such as QINTER, and you have a large number of devices (greater than 300), you run the risk of a subsystem slowdown due to device recovery processing. It is much better to keep a limit on the number of devices allocated to a single subsystem.

It is highly recommended to find a better solution to getting users to the right subsystem rather than using Transfer Job. The next section discusses several techniques that can be used for assigning device names, thus providing the ability to have device names that are associated with users. Once past this hurdle, you can then use workstation entries to get the user to the desired subsystem without using Transfer Job.

### Assign device names

The system has a default naming convention used for display sessions. Today, the most common way to sign-on to a 5250 session is by using Telnet. For example, if you are using iSeries<sup>(TM)</sup> Access to get a sign on display, you are actually using PC5250 which uses Telnet as the underlying communications mechanism. When using Telnet, you get the system default naming convention of QPADEV\* for your device names unless you make changes on your system. This has several significant disadvantages, including:

- You cannot disconnect jobs that use devices that are automatically selected by the system using the default device naming convention (i.e., QPADEV\*).
- It is difficult to manage multiple users or debug intermittent problems when all devices are named QPADEV\*.
- Splitting the devices into multiple subsystems becomes somewhat arbitrary because you cannot subdivide the work based upon type of work or the user doing the work.

- Workstation entries route work by device name, but you may want to route the work by user profile. You cannot make an association between a QPADEV\* and a particular user.

You can make changes on your system to overcome the shortcomings of the system's default behavior by assigning and managing your own device naming convention. There are several ways in which this can be done. All are discussed in the following section. Each approach has its own set of advantages and disadvantages.

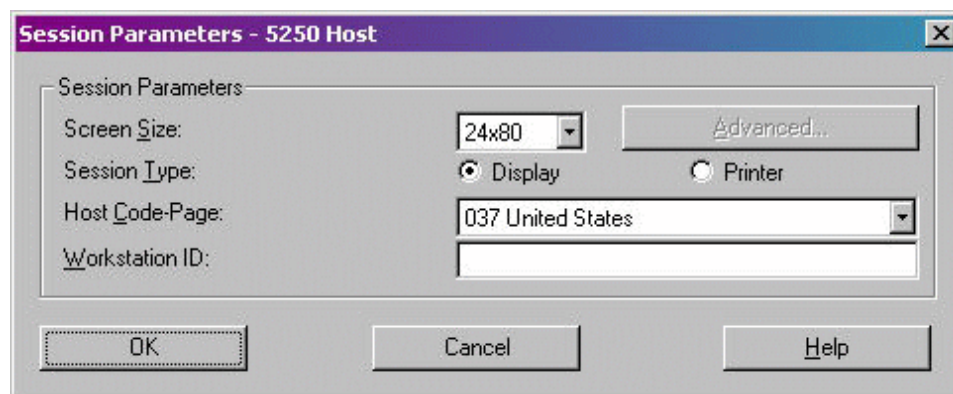
1. Telnet Device Initialization and Termination Exit Points. These exit points provide the ability to assign device names based upon the client signing on to the system. The exit point provides you with the client IP address and the user profile name (along with additional information). You can then 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 Telnet Exit Point documentation provides additional details on these exit points. Technical Studio: Telnet Exit Programs provides sample exit programs to get you started.

The major advantage to using these exit points to manage your device naming convention is that you have central control on the iSeries server for all your clients. The major disadvantage is that it requires programming skills; however the Technical Studio has example exit program source code available that is quite functional.

2. Device Selection Exit Point. This exit point allows you to specify the naming conventions used for automatically created virtual devices and virtual controllers and to specify the automatic creation limit used for the specific 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 Autoconfigure virtual devices ( QAUTOVRT) system value in a more granular 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. See the Device Selection Exit Point documentation for more information.

This exit point gives you the ability to control the default naming conventions used for devices (i.e., QPADEV\*) but it alone will 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, etc.) because it allows you to use different device naming conventions and granular QAUTOVRT management for different access methods.

3. PC5250 (iSeries Access) workstation ID support. You can configure iSeries Access to connect with a specific workstation name. If you select the help button from this panel, the various options for specifying the workstation ID, like generating a new name if the one specified is already in use, are displayed.



The major 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.

```

Start TCP/IP TELNET (STRTCPTELN)

Type choices, press Enter.

ASCII page scroll feature . . . *NO          *NO, *YES
ASCII answerback feature . . . *NONE
ASCII tab stops . . . . . *DFT          0-133, *DFT, *NONE
+ for more values
Coded character set identifier *MULTINAT  1-65533, *MULTINAT...
ASCII operating mode ID . . . *VT220B7 *VT220B7, *VT220B8, *VT100...
Port . . . . . *DFT          1-65534, *DFT
Remote virtual display . . . *DFT          Name, *DFT
Remote user . . . . . *NONE        Name, *NONE, *CURRENT
Remote password . . . . . *NONE

Remote password encryption . . *DES7          *DES7, *SHA1, *NONE
Remote initial program . . . *RMTUSRPRF    Name, *RMTUSRPRF, *NONE
Remote initial menu . . . . *RMTUSRPRF    Name, *RMTUSRPRF, *SIGNOFF
Remote current library . . . . *RMTUSRPRF    Name

```

The major 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 could create a custom version of the STRTCPTELN command to ensure the remote virtual terminal display value and invoke the IBM<sup>(R)</sup> -supplied command.

- You can manually create your virtual controllers and devices. For more information on virtual device creation for Telnet, see the Configure the Telnet Server topic in the iSeries Information Center.

This allows you control over what the names of your controllers and devices are, but it does not provide you the ability to map a specific device to a specific user.

## Manage inactive interactive sessions

Inactive interactive sessions are user jobs that remain signed on, but have not performed any I/O to terminal, have not changed state, or have not used any CPU time during a specified timeframe. These inactive sessions can represent a security exposure if terminal sessions are signed on but left unattended. Fortunately, OS/400 provides the facilities for you to time out inactive interactive sessions.

There are two system values that are used for inactive session management.

- Inactive Time-out Interval (QINACTIV). This system value determines the interval in which the system checks for inactive sessions.
- Inactive Time-out Message Queue (QINACTMSGQ). This system value designates the action that is taken when inactive sessions are timed out.

The system is shipped with the QINACTIV system value set to \*NONE, so by default no sessions are timed out. You will probably want to change the value from \*NONE to some number of minutes. The maximum time-out value is 300 minutes (5 hours). Select a time value that is appropriate for your server usage.

You need to know how the QINACTIV time-out interval works to understand when sessions are timed out. When the time interval expires, all interactive subsystems check the jobs running in their subsystem to determine if they have been idle during the timeframe. If they have been idle, the action defined by the

QINACTMSGQ system value is taken for the inactive job. It is possible for a job to be idle nearly two times the QINACTITV value before the system detects that it has been inactive. This can occur when a job starts shortly after the last inactivity interval ended. For example, if QINACTITV is set to 10 minutes and at 9:10 the system checks for inactive jobs. At 9:12 a new interactive job, NEWJOB, starts but nothing is done in that job. At 9:20 the system checks again for inactive jobs; however, NEWJOB has not been inactive for 10 minutes, it has only been inactive for 8 minutes, so it is not timed out. At 9:30 the system checks again for inactive jobs and this time NEWJOB is timed out, after being inactive for 18 minutes.

As mentioned above, QINACTMSGQ determines the action the system takes on inactive sessions. This system value has three options:

- \*ENDJOB - The inactive jobs are ended
- \*DSCJOB - The inactive jobs are disconnected.

**Note:** Jobs that use QPADEV\* devices cannot be disconnected. In order to sign back on to a disconnected device, you must sign on to the system with the same device description and user profile. However, with system assigned virtual devices it is unlikely to sign on to the same QPADEV\* device again, so the system disallows disconnecting jobs from these devices. To use the disconnect job option you assign device names. See the topic Assign device names earlier in this article.

- Message Queue and Library. A message is sent to the specified message queue for each session that has been inactive. This option allows you to write a program and manage the inactive jobs on your own. This can be very useful if you need more granularity than the system wide behavior that \*ENDJOB or \*DSCJOB give you. For example, you may always want to end JOHN's jobs that have been inactive, MARY's jobs to be disconnected, and BOSS's jobs to always be allowed to remain active.

Inactive device time-out does not apply to QShell or PASE terminal sessions. These sessions never go into a display wait condition so they are never considered inactive and will never be timed out.

## Manage Device Recovery

The Device Recovery Action system value (QDEVRCYACN) and the device recovery action job attribute determine what happens when a device error occurs for an interactive job. The following are the options:

- \*MSG - This option is not recommended due to the potential for security exposures
- \*DSCMSG - The job is disconnected if possible (the job will be ended for jobs with QPADEV\* devices). When you sign on to the device again, you get the option to reconnect to the old job or end the old job. If you reconnect to the old job, an error message is sent to the running application.
- \*DSCENDRQS - The job is disconnected if possible (the job will be ended for jobs with QPADEV\* devices). When you sign on to the device again, you get the option to reconnect to the old job or end the old job. If you reconnect to the old job, a cancel request function is performed.
- \*ENDJOB - The job is ended and a job log is produced.
- \*ENDJOBNOLOG - The job is ended and no job log is generated. This is a good option to take since it avoids the system overhead of creating job logs, particularly if many jobs end at once due to a network problem.

## Determining the IP Address for a Device Description

If you are using Telnet to access the iSeries, you can find out the remote IP address of the user for a device description. This can be done while the job is active at the device.

To do this, use the Retrieve Device Description (QDCRDEVD) API, format DEVD0600.



---

## References and resources

iSeries<sup>(TM)</sup> Information Center

- Work Management
- System Values

Manuals

- V4R5 Work Management 
- Job Scheduler for OS/400 





---

## Disclaimer

Information is provided "AS IS" without warranty of any kind. Mention or reference to non-IBM products is for informational purposes only and does not constitute an endorsement of such products by IBM.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.







Printed in USA