iSeries

Interactive subsystem configuration

Experience Report

iSeries

Interactive subsystem configuration

Contents

Subsystem Configuration 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(R) WebSphere(R) Host Access Transformation Server (HATS), and iSeries(TM) Access for the Web 5250 support.

This experience report includes the following information:

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(R) WebSphere(R) Host Access Transformation Server (HATS), and iSeries(TM) 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 such that you can have an understanding about which users are accessing your system. Thus, 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 SBSD(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 worktation 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 Information Center article about how to create a signon display file.

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 SBSD(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 SBSD(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, refer to the Information Center article about routing entries .

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 SBSD(SBSLIB/INTER1) WRKSTN(QPADEV000*) AT(*SIGNON)
        ADDWSE SBSD(SBSLIB/INTER1) WRKSTN(QPADEV003*) AT(*SIGNON)
        ADDWSE SBSD(SBSLIB/INTER1) WRKSTN(QPADEV006*) AT(*SIGNON)
        ADDWSE SBSD(SBSLIB/INTER1) WRKSTN(QPADEV009*) AT(*SIGNON)
        ADDWSE SBSD(SBSLIB/INTER1) WRKSTN(QPADEV00D*) AT(*SIGNON)
        ADDWSE SBSD(SBSLIB/INTER1) WRKSTN(QPADEV00H*) AT(*SIGNON)
        ADDWSE SBSD(SBSLIB/INTER1) WRKSTN(QPADEV00L*) AT(*SIGNON)
        ```

```
ADDWSE SBSD(SBSLIB/INTER1) WRKSTN(QPADEV00P*) AT(*SIGNON)
ADDWSE SBSD(SBSLIB/INTER1) WRKSTN(QPADEV00S*) AT(*SIGNON)
ADDWSE SBSD(SBSLIB/INTER1) WRKSTN(QPADEV00W*) AT(*SIGNON)
```

- INTER2 would allocate:

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

- INTER3 would allocate:

```
ADDWSE SBSD(SBSLIB/INTER3) WRKSTN(QPADEV002*) AT(*SIGNON)
ADDWSE SBSD(SBSLIB/INTER3) WRKSTN(QPADEV005*) AT(*SIGNON)
ADDWSE SBSD(SBSLIB/INTER3) WRKSTN(QPADEV008*) AT(*SIGNON)
ADDWSE SBSD(SBSLIB/INTER3) WRKSTN(QPADEV00C*) AT(*SIGNON)
ADDWSE SBSD(SBSLIB/INTER3) WRKSTN(QPADEV00G*) AT(*SIGNON)
ADDWSE SBSD(SBSLIB/INTER3) WRKSTN(QPADEV00K*) AT(*SIGNON)
ADDWSE SBSD(SBSLIB/INTER3) WRKSTN(QPADEV00N*) AT(*SIGNON)
ADDWSE SBSD(SBSLIB/INTER3) WRKSTN(QPADEV00R*) AT(*SIGNON)
ADDWSE SBSD(SBSLIB/INTER3) WRKSTN(QPADEV00V*) AT(*SIGNON)
ADDWSE SBSD(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 SBSD(SBSLIB/EAST) WRKSTN(EAST*) AT(*SIGNON)
ADDWSE SBSD(SBSLIB/CENTRAL) WRKSTN(CENTRAL*) AT(*SIGNON)
ADDWSE SBSD(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 SBSD(SBSLIB/PGMR) WRKSTN(PGMR*) AT(*SIGNON)
ADDWSE SBSD(SBSLIB/ORDERENT) WRKSTN(ORDERENT*) AT(*SIGNON)
ADDWSE SBSD(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 SBSD(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 SBSD(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 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 SBSD (QGPL/QINTER) WRKSTN(QPADEV*) AT(*ENTER)
      ```

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

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

   3) And for the third example:

      ```
      ADDWSE SBSD(QGPL/QINTER) WRKSTN(PGMR*) AT(*ENTER)
      ADDWSE SBSD(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” on page 6 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           1-3000
------------------------------------------------------------
```

What this program does:

- Creates up to 999 subsystems using a naming convention of aaaaaaaNNN.
  - 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
                                                                          Start
 Opt    Seq Nbr    Program       Library       Compare Value               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(4) DFT(200) +
                          RANGE(1 3000) 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(4 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(ERROREXIT))
/*                                                                           */
/*****************************************************************************/
/* 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 &NBRSBS) 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    SBSD(&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) +
                         JOBD(*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 * &NBRSBS)) 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 i5/OS(R).

Getting users to a specific subsystem

For interactive sessions, i5/OS 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 i5/OS 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 Information Center articles about routing entries). 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 3000), 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(TM) 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.

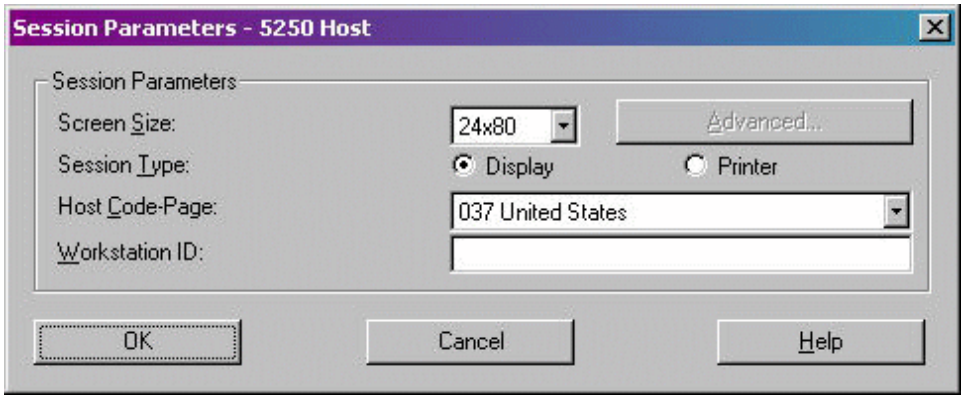
1. Telnet Device Initialization and Termination Exit Points. These exit points provide the ability to assign device names based upon the client signing on to the system. The exit point provides you with the client IP address and the user profile name (along with additional information). You can then 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.

4. i5/OS Telnet Client. Using the i5/OS 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(R) -supplied command.

5. 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, i5/OS 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 (QINACTITV). 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 QINACTITV 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 QINACTITV 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.
- *ENDJOBNOLIST - 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(TM) Information Center
- Work Management
- System Values

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