

## Work Management APIs: Keeping an Eye on Job Activity

[Carsten Flensburg](#)

Wed, 12/18/2013 - 11:15am

Monitor, manage, and interact with subsystems, job queues, and jobs



One thing that fascinates me about APIs is the ways in which they let us create and tailor the tools we need in our daily work to best meet our specific requirements. IBM has invested a significant amount of work and effort into providing GUI work management tools such as [Navigator for i](#). These contemporary offerings include many facilities and a lot of information not available in older, native commands.

So at times, I'm confronted with requests for enhancements of and alternatives to CL commands from people who, in their daily jobs, lack access to the GUI tools, or they want a shortcut to that functionality without having to start Navigator. That's when I turn to the Work Management and User Interface Management (UIM) APIs, which I'll demonstrate in this article.

### A Collection of Past CL Commands

In the past, I've written and published many work management CL commands, so in this article I'll present a collection of these utilities. All of the commands have been updated and enhanced to reflect additions made and changes introduced after their initial release. I've also included a new command not previously published.

The objective of collecting the CL commands in question is twofold. The main purpose is to establish a tool that lets you easily monitor and manage the job streams as jobs are executing in all the subsystems configured on your system. The other purpose is to provide two levels of access to the overview. The first is an interface where you can monitor and interact with the subsystems, job queues, and jobs listed. An alternate interface, which only lets you display the various configuration entities, targets a wider audience that can monitor the system workload as it progresses without compromising operational security and integrity.

For the management-capable version, I provide in a single panel a list of all active subsystems, each specifying a total of currently active jobs and a total of jobs that are currently waiting in the subsystem's attached job queues. If a particular situation in one subsystem catches your eye, the list panel lets you select that subsystem and list all its job queues, again displaying the total jobs currently active from a specific job queue and the total jobs waiting on the queue. The job queue list in turn provides access to a panel that shows all active jobs submitted through the job queue and the jobs waiting on the job queue. The jobs are listed in the order dictated by each job's status and job priority, letting you monitor and predict the job flow in a single view.

I've named the three commands Work with Subsystem Activity (WRKSBSACT), Work with Subsystem Job Queues (WRKSBSJOBQ), and Work with Job Queue Jobs (WRKJOBQJOB). The corresponding display only-capable commands are Display Subsystem Activity (DSPSBSACT), Display Subsystem Job Queues (DSPSBSJOBQ), and Display Job Queue Jobs (DSPJOBQJOB). The Additional Message Information panel discussed in "[Work Management APIs—Putting the Pieces Together](#)" is accessible from both versions. The display version provides access to the Display Job Status (DSPJOBSTS) command and the Display User Jobs (DSPUSRJOB) command, which I've also written, although the former command was previously unpublished.

### Retrieving Job Information

Producing a list and status of all or some of the subsystems configured on an IBM i is easy. You obtain the former by using the Open List of Objects (QGYOLOBJ) API and the latter by using the Retrieve Subsystem Information (QWDRSBSD) API. Likewise, the Open List of Job Queues (QSPOLJBQ) API and the Retrieve Job Queue Information (QSPRJQBQ) API can retrieve the job queue information required to build a list of job queues associated with a subsystem as well as retrieve each job queue's operational status. The heart of the matter very much relies on the Open List of Jobs (QGYOLJOB) API. As Figure 1 shows, the QGYOLJOB API supports a parameter list of 17 parameters, four of which are optional.

Figure 1: Open List of Jobs (QGYOLJOB) API

## Required Parameter Group:

1	Receiver variable	Output	Char(*)
2	Length of receiver variable	Input	Binary(4)
3	Format name	Input	Char(8)
4	Receiver variable definition information	Output	Char(*)
5	Length of receiver variable information	Input	Binary(4)
6	List Information	Output	Char(80)
7	Number of records to return	Input	Binary(4)
8	Sort information	Input	Char(*)
9	Job selection information	Input	Char(*)
10	Size of job selection information	Input	Binary(4)
11	Number of fields to return	Input	Binary(4)
12	Key of fields to be returned	Input	Array(*) of Binary(4)
13	Error Code	I/O	Char(*)

## Optional Parameter Group 1:

14	Job selection format name	Input	Char(8)
----	---------------------------	-------	---------

## Optional Parameter Group 2:

15	Reset status statistics	Input	Char(1)
16	General return data	Output	Char(*)
17	Length of general return data	Input	Binary(4)

Default Public Authority: \*USE

The QGYOLJOB API parameter list is explained in detail in the article “[APIs at Work—with Jobs](#).” The parameter of particular interest for retrieving jobs submitted through a specific job queue is the *Job selection information*. This parameter supports a basic and an enhanced format of selection criteria—named OLJS0100 and OLJS0200, respectively—that lets you narrow the list of jobs returned by the API. You can see the basic list’s offering in Figure 2.

Figure 2: OLJS0100 basic list's offering

Primary job status	ARRAY(*) of CHAR(10)
Active job status	ARRAY(*) of CHAR(4)
Jobs on job queue status	ARRAY(*) of CHAR(10)
Job queue names	ARRAY(*) of CHAR(20)

The enhanced list includes the criteria in Figure 2 and the additional criteria shown in Figure 3.

Figure 3: OLJS0200 enhanced list's offering

Current user profile	ARRAY(*) of CHAR(10)
Server type	ARRAY(*) of CHAR(30)
Active subsystem	ARRAY(*) of CHAR(10)
Memory pool	ARRAY(*) of BINARY(4)
Job type - enhanced	ARRAY(*) of BINARY(4)
Qualified job name	ARRAY(*) of CHAR(26)

As indicated by the *array* key word, you can specify multiple values for each criterion. Consequently, the OLJS0100 and OLJS0200 formats for each criterion contain the offset from the beginning of the structure to the location of the criteria as well as include the count of values provided. This can make the definition of the data structure a bit challenging at first, but after you do the math a couple of times, it all adds up. Here’s an example: according to the API documentation, the OLJS0200 structure’s fixed format includes the 60-byte fixed format from the OLJS0100 format plus 12 4-byte integers holding array offsets and value counts—all in 108 bytes. Because I place the job selection value arrays immediately after the value offset and count section, I use this value as the starting position of the first selection criterion array, and then I add the sum of the preceding criterion array’s sizes to calculate the offset of subsequent selection criterion arrays. Figure 4 shows an example of the OLJS0200 job selection data structure.

Figure 4: OLJS0200 job selection data structure example

```

**-- Selection information:
D OLJS0200          Ds                      Qualified
D  JobNam           10a  Inz( '*ALL' )
D  UsrPrf           10a  Inz( '*ALL' )
D  JobNbr           6a   Inz( '*ALL' )
D  JobTyp           1a   Inz( '*' )
D                   1a
D  OfsPriSts        10i 0 Inz( 108 )
D  NbrPriSts        10i 0 Inz( 2 )
D  OfsActSts        10i 0 Inz( 128 )
D  NbrActSts        10i 0 Inz( 0 )
D  OfsJbqSts        10i 0 Inz( 136 )
D  NbrJbqSts        10i 0 Inz( 0 )
D  OfsJbqNam        10i 0 Inz( 146 )
D  NbrJbqNam        10i 0 Inz( 1 )
D  OfsCurUsr        10i 0 Inz( 166 )
D  NbrCurUsr        10i 0 Inz( 0 )
D  OfsSvrTyp        10i 0 Inz( 176 )
D  NbrSvrTyp        10i 0 Inz( 0 )
D  OfsActSbs        10i 0 Inz( 206 )
D  NbrActSbs        10i 0 Inz( 0 )
D  OfsMemPool       10i 0 Inz( 216 )
D  NbrMemPool       10i 0 Inz( 0 )
D  OfsJobTypeE      10i 0 Inz( 220 )
D  NbrJobTypeE      10i 0 Inz( 0 )
D  OfsJobNamQ       10i 0 Inz( 228 )
D  NbrJobNamQ       10i 0 Inz( 0 )
**
D  PriSts           10a  Dim( 2 )
D  ActSts           4a   Dim( 2 )
D  JbqSts           10a  Dim( 1 )
D  JbqNam           20a  Dim( 1 )
D  CurUsr           10a  Dim( 1 )
D  SvrTyp           30a  Dim( 1 )
D  ActSbs           10a  Dim( 1 )
D  MemPool          10i 0 Dim( 1 )
D  JobTypeE         10i 0 Dim( 1 )
D  JobNamQ          26a  Dim( 1 )

```

To find each criterion array's size, multiply the number of array elements defined for the criterion in the job selection data structure by the length of a single element. For example, if you define two elements for the primary job status criterion located at offset 108, the next criterion will begin at offset 128. I always define a minimum of one element for each criterion. If I don't want to use a criterion, I simply specify a count of zero for that criterion. If I need to specify more values than were previously defined, I add the size of the added elements to all subsequent offsets. I've found that this approach makes it quite easy to work with this somewhat complex parameter because to some extent it documents the correlation between the various elements of the data structure defining it.

The job selection parameter also plays an important role regarding the constraints that apply to the QGYOLJOB API. If you specify one or more values for a job selection criterion, you must request the corresponding job attribute in the array of fields to return, defined by the API's 12th parameter. Basically, this means that for the QGYOLJOB API to select jobs based on a specific job attribute, you must request the API to include that attribute with the job information returned. Based on my experience, this is one of the most common causes for a QGYOLJOB API call to fail, and the requirement isn't explicitly documented in the API manual, although the resulting error message typically will identify the culprit.

## API Programming Results

As for the final outcome of all the API programming efforts, [Figures 5A–E](#) and [Figures 6A–E](#) show the command prompts and list panels, respectively, for some of the commands accompanying this article. To obtain the list options pointing to the Work with Jobs (WRKJOBS) command and the Additional Message Information panel presented in “[Work Management APIs: Putting the Pieces Together](#),” you'll want to download and compile the sources included in that article. In addition, the “How to Compile” section (included with the online version of this article) provides instructions for creating the Work with Jobs (WRKJOBS) command and all its associated commands and objects.

## Find Out More

## Articles at iProDeveloper.com

[“APIs at Work—with Jobs”](#)

[“APIs by Example: Message Handling APIs & Additional Message Info Support”](#)

[“APIs by Example: UIM & Work Management APIs, Part 1”](#)

[“APIs by Example: UIM & Work Management APIs, Part 2”](#)

[“APIs by Example: UIM & Work Management APIs, Part 3”](#)

[“Display Active and Waiting Jobs and Corresponding Job Queues”](#)

[“Displaying User Jobs—But No Changes Allowed with Command DSPUSRJOB”](#)

[“New Command: Display Subsystem Activity—DSPSBSACT”](#)

[“New Command to Display Job Queues by Subsystem”](#)

[“Work Management APIs: Putting the Pieces Together”](#)

[“Carsten's Corner—New Subsystem Entry Commands—Part one”](#)

[“Carsten's Corner—New Subsystem Entry Commands—Part Two”](#)

[“Carsten's Corner—New Subsystem Entry Commands—Part 3”](#)

[“Carsten's Corner—New Subsystem Entry Commands—Part 4”](#)

## IBM i 7.1 Information Center documentation

[Open List of Jobs \(OGYOLJOB\) API](#)

[Open List of Job Queues \(QSPOLJBQ\) API](#)

[Open List of Objects \(QGYOLOBJ\) API](#)

[Register Activation Group Exit Procedure \(CEE4RAGE\) API](#)

[Retrieve Job Queue Information \(OSPRJOBQ\) API](#)

[Retrieve Subsystem Information \(QWDRSBSD\) API](#)

## How to Compile

Below, you'll find instructions for creating the Work with Jobs (WRKJOBS) command and all its associated commands and objects. The following sources are included with the code download available with this article:

CBX156—RPGLE: Work with Subsystem Activity—CCP

CBX156E—RPGLE: Work with Subsystem Activity—UIM Exit Program

CBX156H—PNLGRP: Work with Subsystem Activity—Help

CBX156M—CLP: Work with Subsystem Activity—Build command

CBX156P—PNLGRP: Work with Subsystem Activity—Panel Group

CBX156V—RPGLE: Work with Subsystem Activity—VCP

CBX156X—CMD: Work with Subsystem Activity

CBX157—RPGLE: Work with Subsystem Job Queues—CCP

CBX157E—RPGLE: Work with Subsystem Job Queues—UIM Exit Program

CBX157H—PNLGRP: Work with Subsystem Job Queues—Help

CBX157M—CLP: Work with Subsystem Job Queues—Build command

CBX157P—PNLGRP: Work with Subsystem Job Queues—Panel Group

CBX157V—RPGLE: Work with Subsystem Job Queues—VCP

CBX157X—CMD: Work with Subsystem Job Queues

CBX158—RPGLE: Work with Job Queue Jobs—CCP

CBX158E—RPGLE: Work with Job Queue Jobs—UIM Exit Program

CBX158H—PNLGRP: Work with Job Queue Jobs—Help

CBX158M—CLP: Work with Job Queue Jobs—Build command

CBX158P—PNLGRP: Work with Job Queue Jobs—Panel Group

CBX158V—RPGLE: Work with Job Queue Jobs—VCP

CBX158X—CMD: Work with Job Queue Jobs

CBX965—RPGLE: Display User Jobs—CCP

CBX965E—RPGLE: Display User Jobs—UIM General Exit Program

CBX965H—PNLGRP: Display User Jobs—Help

CBX965L—RPGLE: Display User Jobs—UIM List Exit Program

CBX965M—CLP: Display User Jobs—Build command

CBX965P—PNLGRP: Display User Jobs—Panel Group

CBX965V—RPGLE: Display User Jobs—VCP

CBX965X—CMD: Display User Jobs

CBX966—RPGLE: Display Subsystem Activity—CCP

CBX966E—RPGLE: Display Subsystem Activity—UIM Exit Program

CBX966H—PNLGRP: Display Subsystem Activity—Help

CBX966M—CLP: Display Subsystem Activity—Build command

CBX966P—PNLGRP: Display Subsystem Activity—Panel Group

CBX966V—RPGLE: Display Subsystem Activity—VCP

CBX966X—CMD: Display Subsystem Activity

CBX967—RPGLE: Display Subsystem Job Queues—CCP

CBX967E—RPGLE: Display Subsystem Job Queues—UIM Exit Program

CBX967H—PNLGRP: Display Subsystem Job Queues—Help

CBX967M—CLP: Display Subsystem Job Queues—Build command

CBX967P—PNLGRP: Display Subsystem Job Queues—Panel Group

CBX967V—RPGLE: Display Subsystem Job Queues—VCP

CBX967X—CMD: Display Subsystem Job Queues

CBX968—RPGLE: Display Job Queue Jobs—CCP

CBX968E—RPGLE: Display Job Queue Jobs—UIM Exit Program

CBX968H—PNLGRP: Display Job Queue Jobs—Help

CBX968M—CLP: Display Job Queue Jobs—Build command

CBX968P—PNLGRP: Display Job Queue Jobs—Panel Group

CBX968V—RPGLE: Display Job Queue Jobs—VCP

CBX968X—CMD: Display Job Queue Jobs

CBX265—RPGLE: Display Job Status—CCP

CBX265E—RPGLE: Display Job Status—UIM General Exit Program

CBX265H—PNLGRP: Display Job Status—Help

CBX265L—RPGLE: Display Job Status—UIM List Exit Program

CBX265P—PNLGRP: Display Job Status—Panel Group

CBX265V—RPGLE: Display Job Status—VCP

CBX265X—CMD: Display Job Status

CBX265M—CLP: Display Job Status—Build command

CBX265M—CLP: Work with Jobs—Build commands

To create the above commands and associated objects, compile and run the CBX265M CL program as well as all the other CL programs included in the list, following the instructions in the source header. You'll also find compilation instructions in the respective source headers of the individual sources.

**Source URL:** <http://iprodeveloper.com/application-development/work-management-apis-keeping-eye-job-activity>