

[print](#) | [close](#)

Work Management APIs: Putting the Pieces Together

[Carsten Flensburg](#)

Mon, 10/28/2013 - 4:21pm

Support all list actions and retrieve and identify messages



In previous APIs by Example columns, I've thoroughly covered the Work Management APIs. To add a practical aspect to the explanations, these articles addressed many new CL commands that deal with job information and management. In this article, I'll put all these pieces together to present an updated version of the Work with Jobs (WRKJOBS) command, which in this incarnation provides access to a whole suite of native as well as APIs by Example job commands.

In addition, I'll briefly discuss the programming technique involved in supporting more commands in a User Interface Manager (UIM) list panel than the panel's list options space permits. I'll also present the Work Management API support introduced in recent releases for identifying and handling jobs in a message wait state. Displaying and optionally replying to the message require a bit of coding on your part, but I've included an Additional Message Information panel to relieve most of that burden (for compilation instructions, see the "How to Compile" section below).

Supporting All List Actions

The number of job commands you can access from the Work with Jobs list panel greatly exceeds the space available for the list options in the display panel and, therefore, requires a special technique to support all list option commands and functions. In contrast with the *F24=More keys* functionality, which the UIM handles automatically, there's no corresponding UIM support for *F23=More options*. The method I'll present in this article is based on examples and discussions of the approach contributed by Ed Fischel and the late Simon Coulter, both former IBM associates.

UIM panel groups easily support conditional panel elements such as the option list and associated list actions. You simply specify the name of the condition, declare a condition statement, and indicate the condition name for the condition tag on the list action keyword. Consequently, the list action will appear and be accessible only when the UIM evaluates the condition to be true or not when it displays the panel. A UIM truth table defines certain conditions to be mutually exclusive, letting the UIM avoid reserving panel space for all defined list actions simultaneously.

The challenge in making the *More options* dialog work is keeping all list actions available and active, because the panel displays only some of the list actions. To achieve this, you define all list actions for each condition but specify a text attribute only for those list actions that are to appear in the respective list action sets. [Web Figure 1](#) shows sample panel group source code that demonstrates this technique. Note the following elements:

- The MOROPT variable keeps track of the current state (i.e., which of the two option lists to show). This variable correlates with the CTLRCD parameter structure that communicates between the UIM exit program and the panel group when the value of the MOROPT variable is set.

- The OPTO1 and OPTO2 conditions, which the example defines in the condition statements following the variable and record declaration, are both based on the current value of the MOROPT variable.
- The PNLTT truth table declares that the OPTO1 and OPTO2 conditions are mutually exclusive.
- Two identical sets of UIM list actions are defined, with each set conditioned by OPTO1 and OPTO2, respectively.

The first set defines the text attribute only for the two list actions that will display when the OPTO1 condition evaluates to true; the two subsequent list actions have blank text attributes. Conversely, on the second list action set, which the OPTO2 condition controls, the first list actions have blank text attributes and the two final list actions include a text attribute.

This configuration results in the UIM allowing all list actions to always be responsive while displaying only a subset of list options in the list panel at any given time. In the panel group source code, I define F23 as the *More options* key with an action of calling the UIM exit program. [Figure 1](#) shows the partial source code of that exit program and reveals that toggling the list panel options requires just a little work.

Pressing the F23 key passes control to the exit program, which simply retrieves the CTLRCD parameter structure and the MOROPT variable subfield. The value of the MOROPT variable then increases by one. If the new value exceeds the maximum value of the MOROPT variable, the value resets to zero. Next, the CTLRCD parameter structure is passed back to the panel group and the MOROPT variable's new value causes the displayed list panel options to change, conforming to the associated conditions evaluating to true and not true, respectively.

The maximum value of the MOROPT variable reflects the number of list action sets. In this case, there are two: the first list action set is conditioned by a value of zero; the second is conditioned by a value of one, defining the maximum value in the example in Web Figure 1. This approach lets you easily increase the number of list action sets, given the requirement. Note that if you have more than two list action sets, to reduce complexity you can specify a full set of unconditioned list actions without text and then condition only the list actions that have a text attribute. When calculating the panel layout, the UIM compiler doesn't take into consideration the list actions without text.

Retrieving and Identifying Messages

Some job-related CL commands such as Work with User Jobs (WRKUSRJOB), Work with Submitted Jobs (WRKSBMJOB), and Work with Active Jobs (WRKACTJOB) let you reply to pending inquiry messages for jobs in a message wait status. If you want to replicate this support when designing your own API-based job commands, you might find it interesting to learn that IBM has enhanced the Open List of Jobs (QGYOLJOB) API, the Retrieve Job Information (QUSRJOBI) API, and the Retrieve Thread Attribute (QWTRTVTA) API to include the information necessary to identify and retrieve the inquiry message in question.

The API documentation refers to the job attribute of particular interest in this context as *Message reply*. This attribute defines whether the job is waiting for a reply to a specific message. The field applies only when the active job status or active job status for job ending is MSGW. The possible values include:

- 0: The job currently is not in message wait status.
- 1: The job is waiting for a reply to a message.
- 2: The job is not waiting for a reply to a message.

In plain English, this means the job either isn't in a message wait status (0) or is in a message wait status and is either waiting for a reply to a specific message (1) or waiting for a message to arrive at a message queue as a result of issuing a receive message command or API against that message queue (2). In the event that the message reply attribute has the value 1, the following job attributes let you identify and retrieve the actual inquiry message:

- **Message key:** The key of the message that the active job is waiting for a reply to
- **Message queue name:** The name of the message queue that the active job is waiting to receive a message from
- **Message queue library name:** The name of the library that contains the message queue
- **Library ASP device name:** The name of the auxiliary storage pool (ASP) device description for the ASP containing the library

These job attributes provide the information necessary to retrieve the message identified by the message key from the message queue defined by the ASP, library, and message queue name. Typically, you use the Receive Nonprogram Message (QMHRVCVM) API to perform this task. Once you've retrieved the message details, you format the message text in accordance with the display-panel size and the message-formatting instructions embedded in the message's second-level help text. The latter information is available through the Retrieve Message (QMHRTVM) API.

You must provide a reply line for the user to input the reply to the inquiry message. When input, this reply is forwarded to the waiting message queue via the Send Reply Message (QMHSNDRM) API. The Additional Message Information (AMI) module—included with the accompanying code bundle and ready for you to use in your own utilities—contains the entire message-formatting and reply functionality. Calling the AMI module requires only two parameters: the qualified name of the message queue and the message key.

WRKJOBS Wrap-up

That's it for this discussion of WRKJOBS command basics. To examine what the WRKJOBS command prompt looks like, take a look at Figure 2.

Figure 2: Work with Jobs (WRKJOBS) command prompt

```

Work with Jobs (WRKJOBS)

Type choices, press Enter.

Job name . . . . . *ALL          Name, generic*,
*ALL...
User name . . . . . *ALL          Name, generic*,
*ALL...
Job status . . . . . *ACTIVE      *ACTIVE, *JOBQ,
*OUTQ...
Job type . . . . . *ALL          *ALL, *AUTO,
*BATCH...
Current user . . . . . *NOCHK     Name, *NOCHK

Active status . . . . . *ALL      *ALL, *CNDW, *DEQA,
```

```
*DEQW...
  Completion status . . . . . *ALL          *ALL, *NORMAL,
*ABNORMAL
```

You can narrow the selection of jobs primarily based on generic job name, generic user name, job status, and job type. You can further qualify active jobs by current user and active status and include completed jobs with a completion status of either normal or abnormal. Running the WRKJOBS command for user QTCP on my system produces the list panel in Figure 3.

Figure 3: Work with Jobs (WRKJOBS) list panel

Work with Jobs				WYNDHAMW			11-08-13	
15:16:52								
Type options, press Enter.								
2=Change 3=Hold 4=End 5=Work with 6=Release 7=Display message								
8=Work with spooled files 10=Display job log 12=Work with user jobs...								
Current				Function/				
Opt	Job	User	Job date	Type	---Status---			
Completion								
	QTPPPCTL	QTCP	14-08-13	BCH	ACTIVE	CNDW	PGM-	
QTOCPPPCTL								
	QTSMTPBRSR	QTCP	14-08-13	BCH	ACTIVE	DEQW		
	QTSMTPBRL	QTCP	14-08-13	BCH	ACTIVE	DEQW		
	QTSMTPCLTD	QTCP	14-08-13	BCH	ACTIVE	DEQW		
	QSNMPSA	QTCP	14-08-13	BCH	ACTIVE	DEQW	PGM-	
QNMSARTR								
	QTMSNMPRCV	QTCP	14-08-13	BCH	ACTIVE	TIMW	PGM-	
QTOSRCVR								
	QTSMTPSRVP	QTCP	14-08-13	BCH	ACTIVE	PSRW		
	QTPOP00034	QTCP	14-08-13	BCH	ACTIVE	DEQW		
	QTPOP00034	QTCP	14-08-13	BCH	ACTIVE	DEQW		
	QTPOP00033	QTCP	14-08-13	BCH	ACTIVE	DEQW		
More...								
Parameters or command								
==>								
F3=Exit F4=Prompt F5=Refresh F6=Submit job F9=Retrieve								

```
F11=View 2
  F12=Cancel    F21=Print list    F22=Work with active jobs    F23=More
options
```

The WRKJOBS list panel offers a wide range of job commands—some of IBM origin and others I’ve introduced as part of earlier APIs by Example articles. The following list details the latter:

- [Display Job Open Files \(DSPJOBOPNF\)](#)
- [Display Job SQL Information \(DSPJOBSQLI\)](#)
- [Display Job Log Message \(DSPLOGMSG\)](#)
- [Display Job IP Address \(DSPJOBIPA\)](#)
- [Display Job Screen \(DSPJOBSCN\)](#)
- [Run Job Command \(RUNJOB CMD\)](#)
- [Additional Message Information Panel](#)

In addition to displaying and working with the many job commands available in the panel option list, you can use the F21 key to print the displayed job list and select F22 to run the WRKACTJOB command for the job name that you point the cursor at. You’ll find detailed documentation for the WRKJOBS command and list panel in the accompanying help text panel group by pressing F1. For more information about the CL commands and utilities discussed here, check out the links in the “Find Out More” section.

Find Out More

Articles at iProDeveloper.com

["APIs at Work—with Jobs"](#)

["APIs by Example: Displaying Job Client IP Address and Job Log Information Using APIs"](#)

["APIs by Example: Hidden Job SQL Information Exposed by Retrieve Job Information API"](#)

["APIs by Example: How to Display the Screen of Another Interactive Job"](#)

["APIs by Example: List Open Files API, and the Display Job Open Files Command"](#)

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

["APIs by Example: Use a Work Management API to List Server Jobs"](#)

["APIs by Example: Work with LAN Printers Command"](#)

["Carsten’s Corner—New Work with Remote Output Queue Command"](#)

["Sending Commands to Another Job - Revisited for i5/OS V5R4"](#)

["Use APIs to Monitor and Troubleshoot TCP/IP Processing Jobs"](#)

IBM i 7.1 Information Center documentation

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

[Retrieve Job Information \(QUSRJOBI\) API](#)

[Retrieve Thread Attribute \(QWTRTVTA\) API](#)

[List Open Files \(QDMLOPNF\) API](#)

[Change Job Interrupt Status \(QWCCJITP\) API](#)

[Call Job Interrupt Program \(QWCJBITP\) API](#)

[Open List of Job Log Messages \(QGYOLJBL\) API](#)

[Receive Nonprogram Message \(QMHRCVM\) API](#)

[Retrieve Message \(QMHRTVM\) API](#)

[Send Program Message \(QMHSNDPM\) API](#)

[Send Reply Message \(QMHSNDRM\) API](#)

How to Compile

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

CBX232—RPGLE: Work with Jobs—CCP

CBX232E—RPGLE: Work with Jobs—UIM General Exit Program

CBX232H—PNLGRP: Work with Jobs—Help

CBX232L—RPGLE: Work with Jobs—UIM List Exit Program

CBX232P—PNLGRP: Work with Jobs—Panel Group

CBX232V—RPGLE: Work with Jobs—VCP

CBX232X—CMD: Work with Jobs

CBX232M—CLP: Work with Jobs—Build command

CBX209—RPGLE: Additional Message Information

CBX209E—RPGLE: Additional Message Information—UIM Exit Program

CBX209H—PNLGRP: Additional Message Information—Help

CBX209P—PNLGRP: Additional Message Information—Panel Group

CBX227—RPGLE: Display Job Open Files—CPP

CBX227E—RPGLE: Display Job Open Files—UIM Exit Program

CBX227H—PNLGRP: Display Job Open Files—Help

CBX227P—PNLGRP: Display Job Open Files—Panel Group

CBX227X—CMD: Display Job Open Files

CBX227M—CLP: Display Job Open Files—Build command

CBX228—RPGLE: Display Job SQL Information—CPP

CBX228E—RPGLE: Display Job SQL Information—UIM Exit Program

CBX228H—PNLGRP: Display Job SQL Information—Help

CBX228P—PNLGRP: Display Job SQL Information—Panel Group

CBX228X—CMD: Display Job SQL Information

CBX228M—CLP: Display Job SQL Information—Build command

CBX230—RPGLE: Display Job Log Message

CBX230H—PNLGRP: Display Job Log Message—Help

CBX230X—CMD: Display Job Log Message

CBX230M—CLP: Display Job Log Message—Build command

CBX231—RPGLE: Display Job IP Address

CBX231H—PNLGRP: Display Job IP Address—Help

CBX231X—CMD: Display Job IP Address

CBX231M—CLP: Display Job IP Address—Build command

CBX256H—PNLGRP: Display Job Screen—Help

CBX256V—RPGLE: Display Job Screen—VCP

CBX256X—CMD: Display Job Screen

CBX2561—RPGLE: Display Job Screen—CPP

CBX2562—RPGLE: Display Job Screen—Exit Program

CBX256M—CLP: Display Job Screen—Build command

CBX975H—PNLGRP: Run Job Command—Help

CBX975X—CMD: Run Job Command

CBX9751—RPGLE: Run Job Command—CPP

CBX9752—RPGLE: Run Job Command—Exit Program

CBX975M—CLP: Run Job Command—Build command

CBX264M—CLP: Work with Jobs—Build commands

To create the above exit programs, compile and run the CBX264M CL program, 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-putting-pieces-together>