

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

APIs by Example: Monitor Batch Job Utility

[System iNetwork Programming Tips Newsletter](#)

[Carsten Flensburg](#)

Carsten Flensburg

Thu, 03/17/2005 (All day)

This week's API by Example presents a utility that is designed to monitor active batch jobs and report any active jobs that are currently either held or waiting for a reply to a specific message. I named it "The Monitor Batch Jobs (MONBCHJOB)" command.

The heart of the utility is the QGYOLJOB (Open List of Jobs) API, which, as its name reveals, is one of the Open List APIs. These APIs return the same type of information as regular List APIs. The difference is the standard List APIs output their results to a user space, whereas the Open List APIs spawn a separate job to do the work and you retrieve the data from that job using the Get List Entry (QGYGTLE) API.

Although I could've written the MONBCHJOB utility with the normal List APIs, I chose the Open List APIs to improve performance. Since the information retrieval is an asynchronous process, your program can start processing the returned list entries even before the list is complete.

For large lists, this can substantially improve the performance of the list processing program. When list processing is complete, running the Close List (QGYCLST) API is essential since it performs clean-up activities and frees up all of the resources used by the Open List API.

Another advantage of many of the Open List APIs is a list sort specification parameter that lets you specify the sort order of the retrieved list. This sort facility is based on the same very flexible methodology that's employed by the Sort (QLGSORT) and Sort Input/Output (QLGSRTIO) APIs.

The Sort APIs are described in the National Language Support APIs section of the Information Center, which can be found at the following link:

<http://publib.boulder.ibm.com/infocenter/iseres/v5r4/index.jsp?topic=/apis/nls1.htm>

IBM originally designed the Open List APIs to support requirements of client/server applications, but the usability is not restricted to this type of programming constructs only, as hopefully this week's utility will demonstrate.

Open list APIs were also described in the article entitled "Check Job Queue Status" in the July 29, 2004, issue of this newsletter:

<http://systeminetwork.com/article/apis-example-check-job-queue-status>

The general documentation for the Process Open List APIs is documented here:

<http://publib.boulder.ibm.com/infocenter/iseres/v5r4/index.jsp?topic=/apis/misc1b.htm>

The specific Open List APIs are always documented together with the other APIs covering the same area. In this case, the QGYOLJOB API documentation is located in the Work Management APIs section:

<http://publib.boulder.ibm.com/iseres/v5r2/ic2924/index.htm?info/apis/qgyoljob.htm>

Now on to the MONBCHJOB command -- this is what the command prompt looks like:

```

= = = = =
                          Monitor Batch Jobs (MONBCHJOB)
Type choices, press Enter.
Subsystem . . . . . Name, *ALL
Job queue . . . . . *ALL Name, *ALL
  Library . . . . . Name, *LIBL, *CURLIB
Batch job status . . . . . *BOTH *BOTH, *HELD, *MSGW
Monitor duration:

  Time interval . . . . . *ONCE 1-1440 minutes, *ONCE
  Count limit . . . . . 1-9999, *DAY, *WEEK
Monitor job description . . . *USRPRF Name, *USRPRF
  Library . . . . . Name, *LIBL, *CURLIB
Message queue to be notified . *SYSOPR Name, *SYSOPR, *NONE
  Library . . . . . Name, *LIBL, *CURLIB
Command to execute . . . . .

Command variables:

  Variable . . . . . *PARMSTR *PARMSTR, *JOBNAME...

  Variable name . . . . . Name

      + for more values

= = = = =

```

I have included a help panel group to explain both the command in general and the individual command parameters. Here's a brief overview for your convenience:

```

Subsystem - Defines the subsystem whose jobs should be monitored.
            The special value *ALL is allowed.

Job queue - Limits the batch job monitor to jobs submitted to the
            specified job queue. Special value *ALL is allowed.

Batch job
Status    - The active job status to monitor for, including *HELD
            and *MSGW. Special value *BOTH allows monitoring of
            both statuses.

Monitor
Duration  - Specifies the number of cycles that the job monitor
            should perform and the interval in minutes between
            each monitor run.

Monitor
JobD      - When you run the MONBCHJOB command, a monitor job is
            submitted with the Job Description specified here.

Message
Queue     - This message queue will receive an informational
            message if a monitor event is detected.

Command   - Specify a command to be run if a monitor event is

```

detected. Variable names can be specified as this command's parameters as described below.

Command

Variables - A list of available substitution parameters, including the found job's job name, user profile, job number, active status, subsystem name, job queue name and latest job log message. The special value *PARMSTR results in a data structure that can be used as a CALL command's single parameter, as in this example:

```
MONBCHJOB SBS( *ALL )
          JOBQ( *ALL )
          STATUS( *BOTH )
          MONITOR( 10 *DAY )
          JOBQ( *USRPRF )
          MSGQ( *SYSOPR )
          CMD( CALL PGM( CBX132T ) PARM( ))
          CMDVAR(( *PARMSTR ))
```

Prior to running the above command, the batch job monitor will replace the PARMS parameter name with the *PARMSTR parameter data structure. I have included the CBX132T sample program for you as a starting point to build your own batch job monitor exit program.

Note that all commands are run directly by the batch job monitor. To prevent that the batch job monitor job itself is affected by a possible exception in the exit program, global error monitoring is important.

The following command will send a text message containing job name and job status to the user profile that is running the found job:

```
MONBCHJOB SBS(NOVASYS)
          MONITOR( 1 3 )
          MSGQ( *NONE )
          CMD( SNDMSG MSG('Job is currently .')
              TOUSR())
          CMDVAR(( *JOBNAME ) ( *JOBUSER )
              ( *JOBSTS ))
```

Please refer to the command help text for further details.

The MONBCHJOB command includes the following sources:

```
CBX1321 -- Initial command processing program submitting the
          monitor job.

CBX1322 -- Main command processing program performing the monitor
          job.

CBX132V -- Command validity checking program.

CBX132H -- Command help text panel group.

CBX132X -- Command definition source member.
```

```
CBX132T -- Sample monitor job exit program.
```

Compilation instructions are found in the source headers.

This article demonstrates the following APIs:

Open List of Jobs (QGYOLJOB) API:

<http://publib.boulder.ibm.com/iserics/v5r2/ic2924/index.htm?info/apis/qgyoljob.htm>

Open List of Job Log messages (QGYOLJBL) API:

<http://publib.boulder.ibm.com/infocenter/iserics/v5r3/index.jsp?topic=/apis/QGYOLJBL.htm>

Get List Entries (QGYGTLE) API:

<http://publib.boulder.ibm.com/infocenter/iserics/v5r3/index.jsp?topic=/apis/qgygtle.htm>

Close List (QGYCLST) API:

<http://publib.boulder.ibm.com/iserics/v5r1/ic2924/index.htm?info/apis/qgyclst.htm>

Retrieve User Information (QSYRUSRI) API:

<http://publib.boulder.ibm.com/infocenter/iserics/v5r4/index.jsp?topic=/apis/qsyrusri.htm>

Process Commands (QCAPCMD) API:

<http://publib.boulder.ibm.com/infocenter/iserics/v5r4/index.jsp?topic=/apis/qcapcmd.htm>

Send Nonprogram Message (QMHSNDM) API:

<http://publib.boulder.ibm.com/infocenter/systems/index.jsp?topic=/apis/QMHSNDM.htm>

Send Program Message (QMHSNDPM) API:

<http://publib.boulder.ibm.com/infocenter/iserics/v5r4/index.jsp?topic=/apis/QMHSNDPM.htm>

You can retrieve the source code for this API example from

www2.systeminetwork.com/code/clubtechcode/monbatchjob.zip .

The above article was written by Carsten Flensburg. If you have any questions, you can contact Carsten at <mailto:flensburg@novasol.dk> .

Source URL: <http://iprodeveloper.com/rpg-programming/apis-example-monitor-batch-job-utility>