


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

APIs by Example: Data Queue APIs and CL Commands, Part 3

[System iNetwork Programming Tips Newsletter](#)

[Carsten Flensburg](#)

Carsten Flensburg

Thu, 01/18/2007 (All day)

Today, my addition of new data queue CL commands continues. Taking advantage of data queues in application development often requires peeking at the data queue content to verify or debug the data exchange between your programs. This need inspired the Display Data Queue Entry (DSPDTAQE) command.

The DSPDTAQE command is based on the nondestructive data queue entry retrieval that the Retrieve Data Queue Message (QMHRDQM) API performs. Using this API, you can retrieve as many data queue entries as your receiver variable can hold, without actually removing the data queue entries from the data queue.

On a side note, the nondestructive data queue entry retrieval capability has also been added to the original Receive Data Queue (QRCVDTAQ) API, but this API retrieves only one data queue entry at a time and is therefore of no use if you want to read all data queue entries without actually removing them.

Currently, no IBM-provided command displays a data queue's contents. You could use the Dump Object (DMPOBJ) command to generate an object dump of the data queue, including the queue entries, but analyzing the resulting spooled file can be challenging and time-consuming. Nevertheless, the DMPOBJ command does the job if no other alternative is available.

However, I hope that you find that the DSPDTAQE command is a viable alternative to the DMPOBJ command. Here's how the command prompt looks:

```

                                Display Data Queue Entries (DSPDTAQE)
Type choices, press Enter.
Data queue . . . . .
  Library . . . . . *LIBL      Name, *LIBL, *CURLIB
  Sequence . . . . . *DTAQDFN  *DTAQDFN, *DTAQREV,
*BYKEY...
Select queue entries:
  Relational operator . . .      *GT, *LT, *NE, *EQ, *GE,
*LE
  Key value . . . . .
  Key length . . . . . *DTAQDFN  1-256, *DTAQDFN

```

Running the DSPDTAQE command against the QSYSDTAQ data queue on my system produces a panel somewhat similar to this one:

```

                                Display Data Queue Entries

WYNDHAMW

                                14-01-07

16:53:00
  Data queue . . . :   QSYSDTAQ                Sequence . . . :
*KEYED
  Library . . . :   QSYS                        Entry count . . :   4230

Type options, press Enter.
  5=Display entry data   6=Display entry key
Opt  Entry data
  *JOBNOTIFY01          QSDSNDCFG      QSVCDRCTR 813235QSVCDRCTR
QSVCDRCT
  *JOBNOTIFY01          QHTTP          QTMHHTTP  817638QZHBHTTP
QHTTPSVR
  *JOBNOTIFY01          QTPOC00095     QTCP       817851QSYSNOMAX QSYS
  *JOBNOTIFY01          QTPOP00096     QTCP       817852QSYSNOMAX QSYS
  *JOBNOTIFY01          QHTTP          QTMHHTTP  818007QZHBHTTP
QHTTPSVR
  *JOBNOTIFY01          QHTTP          QTMHHTTP  818209QZHBHTTP
QHTTPSVR
  *JOBNOTIFY01          QDIALLOCAL      QSNADS     826690QSNADS     QGPL
  *JOBNOTIFY01          QDIAINDUSR      QSNADS     826691QSNADS     QGPL
  *JOBNOTIFY01          QDIAHSTPRT     QSNADS     826692QSNADS     QGPL

More...
Command
===>
F3=Exit      F4=Prompt   F5=Refresh   F9=Retrieve   F11=Entry key
F12=Cancel   F17=Top      F18=Bottom

```

As usual, the cursor-sensitive online help text that accompanies the utility explains both the command and the display panel in more detail. However, let me briefly explain the panel displayed when you select option 5 or 6, *Display entry data* and *Display entry key*, respectively. When you choose one of these, the initial panel displayed shows you the data or key in character format. Pressing F11 shows you the same data or key in hexadecimal format.

I added a blank line between all lines in the character format. Further, the hexadecimal format shows the hex nibbles top-down, as in the following example:

```

Character data
*...+....1....
*JOBNOTIFY01

Hexadecimal data
*...+....1....

```

```
5DDCDDECCEFF03
C1625639680100
```

In character format, the second byte is the letter 'J'. The equivalent hexadecimal value is 'D1', but the nibble '1' is below the nibble 'D'. This is what I refer to as top-down, as opposed to left-right, in which the 'D' would be followed by the '1', as you might have expected otherwise. This layout ensures an exact overlay of the two formats and makes it easier to translate between character and hexadecimal format if you use F11 to toggle between the two.

In addition to offering an easy facility to view a data queue's content, the DSPDTAQE utility also conveniently provides examples of calling all the data queue APIs.

As I mentioned initially, the QMHRDQM API is the utility engine that provides access to the data queue entries and key values as well as other entry attributes. I further use the Retrieve Data Queue Description (QMHQRDQD) API to get information about data queue type, whether sender information is stored with the data queue entries, and the data queue entry and key value length. These are all crucial pieces of information that the utility needs to process the data queue and its entries correctly.

Because I'm using a User Interface Manager (UIM) panel group list to display the data queue entries and their associated attributes, and because each data queue entry potentially could have a size of up to 64,512 bytes, I also need to take into account the UIM list size limit of 16 MB. Though only the first 64 bytes of each entry is displayed on the Display Data Queue Entry panel, option 5 on the list panel lets you display the full data queue entry. An easy approach would be to store the full data queue entry in the list and simply retrieve it from the list and display it whenever option 5 is selected. But to reduce each UIM list entry's size, I store only 256 bytes of each data queue entry in the panel group list. If the data queue entry is bigger than 256 bytes, I store the full entry in a temporary storage data queue, using an assigned data queue entry number as the entry key. I do this work with the Send to a Date Queue (QSNDDTAQ) API.

The entry number is stored in the panel group list, as is all other data queue entry information. Then, to retrieve the full data queue entry before displaying it, I use the QRCVDTAQ API, specifying the entry number as key. And finally, whenever the list needs to be rebuilt, the Clear Data Queue (QCLRDTAQ) API ensures that the temporary storage data queue is empty, before any new entries arrive.

To get the whole picture, I suggest that you take a close look at the source code. You could easily extract the data queue API prototypes and calls therein and use them in your own programs. Another option would be to wait for the final installment of this article series, in which I will present a simple example, showing you the basic details of using data queue APIs in an application context.

In the next installment (which is not the final one), I'll wrap up the data queue CL command enhancements that I've introduced so far. To do so, I present an alternate Work with Data Queues command, providing access to the new data queue CL commands that I previously presented in this column.

This APIs by Example includes the following sources:

```
CBX167  -- Display Data Queue Entries - CPP
CBX167E -- Display Data Queue Entries - UIM General Exit Program
CBX167H -- Display Data Queue Entries - Help
CBX167L -- Display Data Queue Entries - UIM List Exit Program
```

```
CBX167P -- Display Data Queue Entries - Panel Group
CBX167V -- Display Data Queue Entries - VCP
CBX167X -- Display Data Queue Entries

CBX167M -- Display Data Queue Entries - Build Command
```

To create all these objects, compile and run CBX167M. Compilation instructions are in the source headers, as usual.

The previous installments of this article series:

Data Queue APIs and CL Commands, Part 1

<http://www.systeminetwork.com/article.cfm?id=53542>

Data Queue APIs and CL Commands, Part 2

<http://www.systeminetwork.com/article.cfm?id=53685>

This article demonstrates the following data queue APIs:

Retrieve Data Queue Description (QMHQRDQD) API:

<http://publib.boulder.ibm.com/infocenter/iserics/v5r3/topic/apis/qmhqrdqd.htm>

Send Date Queue (QSNDDTAQ) API:

<http://publib.boulder.ibm.com/infocenter/iserics/v5r3/topic/apis/qsnddtaq.htm>

Receive Date Queue (QRCVDTAQ) API:

<http://publib.boulder.ibm.com/infocenter/iserics/v5r3/topic/apis/qrcvdtaq.htm>

Clear Data Queue (QCLRDTAQ) API:

<http://publib.boulder.ibm.com/infocenter/iserics/v5r3/topic/apis/qclrdaq.htm>

Retrieve Data Queue Message (QMHRDQM) API:

<http://publib.boulder.ibm.com/infocenter/iserics/v5r3/topic/apis/qmhrdqm.htm>

All data queue APIs are documented here:

<http://publib.boulder.ibm.com/infocenter/iserics/v5r3/topic/apis/obj2.htm>

[Retrieve the source code for this API example.](#)

Source URL: <http://iprodeveloper.com/rpg-programming/apis-example-data-queue-apis-and-cl-commands-part-3>