

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

APIs by Example: New Data Queue Attributes and New Data Queue Command

[System iNetwork Programming Tips Newsletter](#)

[Carsten Flensburg](#)

Carsten Flensburg

Thu, 10/28/2010 (All day)

I use data queues for many purposes and in many mission-critical applications, so whenever IBM announces additions and changes to data queues, I pay attention. With release 6.1, two new data queue attributes were added, and one new data queue API was introduced. The new Change Data Queue (QMHQCDQ) API was partially prompted by the *Enforce data queue locks* attribute added, because the QMHQCDQ API is currently the only available option when it comes to setting this attribute; you cannot specify it when you create a data queue. I cover more about the *Enforce data queue locks* attribute later in this article.

The QMHQCDQ API currently supports changing one more data queue attribute, the *Automatic reclaim* attribute defining whether the allocated storage of the data queue should be reclaimed whenever the last entry in the data queue is removed. This attribute is, however, also available when you create a data queue, but you now have the option of changing it, without having to re-create the data queue. The second attribute added to data queues at release 6.1 is a timestamp registering the point in time the data queue last had its storage automatically reclaimed. These new data queue features inspired me to make corresponding improvements to my previously published data queue commands as well as create the new Change Data Queue (CHGDTAQ) CL command.

The *Enforce data queue locks* attribute lets you ensure that object locks obtained using the Allocate Object (ALCOBJ) command are honored by all IBM-supplied data queue operations, being provided by means of Data Queue APIs and Queue MI instructions and built-ins. At releases prior to 6.1, you could control access to data queues within an application in terms of application design. By attempting to place an object lock on a data queue object prior to performing an exclusive data queue operation, you would discover whether a lock had already been obtained and be able to act appropriately in order to serialize or abandon access to the data queue.

Nothing would, however, prevent you from performing a data queue operation against the data queue without obtaining an object lock first. So whether such an attempt was made on purpose or by mistake, the object lock would not cause it to fail, and therefore it would succeed if no formal errors were encountered. The new enforce locks attribute, however, lets you ensure that object locks are not circumvented when native data queue operations are performed. Setting this attribute to *YES causes the data queue operations to check for lock conditions on the data queue object prior to carrying out the operation. Apart from backward compatibility, the reason this behavior is not simply implemented by default is of course the performance penalty incurred by the object lock condition check.

So for performance-critical and high-volume transaction applications, careful benchmark tests should be included early on in the application design phase, in case the new *Enforce data queue*

locks attribute is part of your design. Here's an overview of the type of object lock required in order to perform various data queue operations with the new attribute set to *YES:

```
*SHRUPD: Clear Data Queue (QCLRDTAQ) API
          Receive Data Queue (QRCVDTAQ) API
          Send Data Queue (QSNDDTAQ) API
          Enqueue (ENQ) MI instruction
          Dequeue (DEQ) MI instruction

*SHRRD:  Retrieve Data Queue Description (QMHQRDQD) API
          Retrieve Data Queue Message (QMHRDQM) API
          Materialize Queue Attribute (MATQAT) MI instruction
          Materialize Queue Message (MATQMSG) MI instruction

*SHRUPD: Change Data Queue (QMHQCDQ) API
          (irrespective of data queue locks being enforced)!
```

As I mentioned earlier, the only option available to set the new enforce locks attribute is the release 6.1—introduced Change Data Queue (QMHQCDQ) API. To accommodate a certain level of convenience, I therefore decided to build a Change Data Queue (CHGDTAQ) CL command based on the aforementioned API. In addition to the enforce locks attribute, the QMHQCDQ API also supports the *Automatic reclaim (data queue storage)* attribute added by IBM quite a while ago, to allow the data queue object to "shrink" to its initial size whenever it is emptied (i.e., the last data queue entry is removed).

Prior to the addition of said attribute, many shops simply re-created their data queues each time an application was started, because at that point this was the only way to get rid of excessive data queue object storage allocation. Should your data queues still suffer from this shortcoming, now you have the option of correcting it without having to re-create the data queue. The CHGDTAQ CL command to the rescue:

Change Data Queue (CHGDTAQ)			
Type choices, press Enter.			
Data queue		Name	
Library	*LIBL	Name, *LIBL,	
*CURLIB			
Automatic reclaim	*SAME	*SAME, *NO, *YES	
Enforce data queue locks	*SAME	*SAME, *NO, *YES	

Initially the CHGDTAQ CL command prompt displays only the data queue parameter, but once you enter a qualified data queue name and press Enter, the current values of the two data queue attributes available for change are retrieved by the command's prompt override program. You can therefore also use the CHGDTAQ CL command to verify the current attribute setting of a given data queue object. I have included a lot of the API manual's comprehensive information and explanations in the online help text panel group, in case any questions or uncertainty should arise when working with the CHGDTAQ CL command prompt.

In the event that you're unfamiliar with the technique involved in creating a command prompt override program (POP), I've included a brief description of the steps involved in adding a POP to a command definition.

1. You identify the command parameters required to retrieve the parameter values necessary. Each identified parameter is called a Key parameter. For the CHGDTAQ command, this is simply the qualified name of the data queue. Had it been the Change Object Description (CHGOBJD) command, for example, you would need to also include the object type.
2. For each identified key parameter, you'll want to add the Keyparm(*YES) attribute to the parameter definition, as in the following example:

```
Parm          DTAQ          Q0001          +
Min( 1 )      +
  Keyparm( *YES )          +
Prompt( 'Data queue' )
```

3. In the POP, you then define the program's parameter structure accordingly; here's how it's defined for the CHGDTAQ command in the CBX220O POP:

```
Parameters:

PxCmdNam_q  Char(20)          INPUT  Qualified command name

PxKeyPrm1   Parm definition  INPUT  Key parameter
identifying the data
attribute
queue to retrieve
information about.

PxCmdStr    VarChar(32674)   OUTPUT  The formatted command
prompt string
attribute
specified data queue
to the command
processor.
```

Note that there will be a PxKeyPrm[n] parameter for each identified key parameter. The data type and size or precision of each parameter equals the corresponding command parameter. Each key parameter is passed in the order they are defined in the command definition source.

At this point it then rests with the POP to retrieve the needed parameter value information as well as format and return the command prompt string to the command processor. To format the command prompt string, you use selective prompt characters, such as, for example, ?? or ?<, which prefix each command keyword containing the parameter value. The main difference between ?? and ?< is the way the command processor handles the parameter value.

- When specifying ?? the actual value of the parameter is always returned to the command processing program (CPP).
- When specifying ?<, however, only values explicitly entered on the command prompt are returned. If nothing is entered, the parameter's default value is returned to the CPP.

Defining the command's default parameter values as *SAME, using the ?< prompt characters allows me to detect whether anything was entered and only perform the change operation if that is actually the case, being the case only if the CPP receives a parameter value different from *SAME. I've included a link below to explain selective prompt characters in more detail. Here's an example of how the command prompt string would look for the CHGDTAQ command:

?

The above command prompt string would cause the CHGDTAQ command to display the following full command prompt:

```

                                Change Data Queue (CHGDTAQ)

Type choices, press Enter.

Data queue . . . . . WEB001DQ      Name
Library . . . . . *LIBL      Name, *LIBL,
*CURLIB
Automatic reclaim . . . . . *YES      *SAME, *NO,
*YES
Enforce data queue locks . . . . . *NO      *SAME, *NO,
*YES
```

The enforce locks attribute as well as an *Automatic reclaim timestamp* has also been added to the API output data structure of the Retrieve Data Queue Description (QMHQRDQD) API's output format RDQD0100. This timestamp designates the point in time the data queue last had its allocated storage automatically reclaimed. Both new data queue attributes being implemented by IBM at release 6.1, have prompted me to update a couple of earlier published data queue CL commands to reflect these enhancements. The Display Data Queue Description

(DSPDTAQD) CL command's display and print panels have had the new attributes added, as demonstrated below by an example showing the command's display panel:

```

                                Display Data Queue Description

WYNDHAMW                                                                22-
10-10  12:15:56
Data queue . . . . . :   WEB001DQ           Type . . . . . :
*STD
Library  . . . . . :   QGPL

Maximum entry length . . . . . :   64512

Force to auxiliary storage . . . :   *NO

Sequence . . . . . :   *FIFO

Include sender ID . . . . . :   *NO

Automatic reclaim . . . . . :   *YES      2010-10-22-
12.01.05.477976
Enforce data queue locks . . . . :   *NO

Queue size:

Maximum number of entries . . . :   *MAX2GB

Initial number of entries . . . :   16

Queue entries:

Current number . . . . . :   0

Current allocated . . . . . :   16

Maximum allowed . . . . . :   33104

More...
```

The automatic reclaim timestamp is located to the right of the *Automatic reclaim* attribute and is shown only if an automatic reclaim has occurred. Otherwise blanks will appear in that location. The *Enforce data queue locks* attribute is displayed right below the automatic reclaim ditto. The display panel and all fields shown are as always explained in the cursor-sensitive help text associated with the display. Just point the cursor to the area or field of interest and press function key F1 to access the help text provided.

The final update I've included with this article is adding the new CHGDTAQ CL command as option 2 on the Work with Data Queues (WRKDTAQ2) command's list panel. The revised source members are all included in the utility source zip file at the end of this article.

This APIs by Example includes the following sources:

```
CBX220  -- RPGLE  -- Change Data Queue - CPP
CBX220H -- PNLGRP -- Change Data Queue - Help
CBX220O -- RPGLE  -- Change Data Queue - POP
CBX220V -- RPGLE  -- Change Data Queue - VCP
CBX220X -- CMD    -- Change Data Queue

CBX220M -- CLP    -- Change Data Queue - Build command
```

To create all these objects, compile and run the CBX220M program, following the instructions in the source header. You'll also find compilation instructions in the respective source headers.

The following previously published sources for the Display Data Queue Description (DSPDTAQD) and Work with Data Queues (WRKDTAQ2) commands have been updated in order to add support for the two new data queue attributes and the new CHGDTAQ command, respectively:

```
CBX165  -- RPGLE  -- Display Data Queue Description - CPP
CBX165H -- PNLGRP -- Display Data Queue Description - Help
CBX165P -- PNLGRP -- Display Data Queue Description - Panel Group
CBX165V -- RPGLE  -- Display Data Queue Description - VCP

CBX168H -- PNLGRP -- Work with Data Queues - Help
CBX168P -- PNLGRP -- Work with Data Queues - Panel Group
```

These sources are all included in the utility source zip file provided at the end of this article, and compilation instructions are found in the respective source headers. As for the remaining DSPDTAQD and WRKDTAQ2 command sources, you'll find them included with the Data Queue APIs and CL Commands article series part 1 and part 4, respectively, for which links are included below.

IBM documentation:

[Using a prompt override program](#)

[Using selective prompting for CL commands](#)

Related articles:

[APIs by Example: Data Queue APIs and CL Commands, Part 1](#)

[APIs by Example: Data Queue APIs and CL Commands, Part 2](#)

[APIs by Example: Data Queue APIs and CL Commands, Part 3](#)

[APIs by Example: Data Queue APIs and CL Commands, Part 4](#)

[APIs by Example: Data Queue APIs and CL Commands, Part 5](#)

This article demonstrates the following data queue APIs:

[Change Data Queue \(QMHQCDQ\) API](#)

[Retrieve Data Queue Description \(QMHQRDQD\) API](#)

[Using Data Queue APIs](#)

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

Source URL: <http://iprodeveloper.com/rpg-programming/apis-example-new-data-queue-attributes-and-new-data-queue-command>