

APIs by Example: List Job Object Locks

[System iNetwork](#)

System iNEWS Staff

Thu, 07/15/2004 (All day)

In this installment of APIs by Example, Carsten Flensburg demonstrates the Retrieve Job Locks (QWCRJBLK) API. Two sample programs have been provided for this article.

The first sample program is named CBX504T. It's a straightforward example of calling the second sample program.

The second sample program is named CBX504, and it's the one that makes the call to the QWCRJBLK API. There are two parts of this: The way that memory is allocated, and the way that the returned list is processed. I'll explain both parts in this article.

When you call a program, one of the very first things that the program does is ask the operating system for a place in memory that it can store each of its variables. The exception to this rule is that any variable, data structure, or other definition that's declared with the BASED keyword does not get memory allocated to it. The system expects that you will either manually allocate memory for the BASED variables or that you will point them at another valid area of memory.

In this situation, a receiver variable is needed that will be large enough to store all of the objects locked for the current job. How do you know how much memory to ask for when you have no way of knowing how many objects are locked?

The program starts out by asking the operating system for 10k of memory using the following two lines of code:

```
C          Eval      ApiRcvSiz   = 10240
C          Eval      pLstHdr     = %Alloc( ApiRcvSiz )
```

Since the JBLK0100 data structure is based on the pLstHdr pointer, that data structure is now able to store up to 10k of information. When the QWCRJBLK API is called, the ApiRcvSiz variable is passed to it to tell the API how much space is available.

The following code runs after the memory is allocated:

```
C          DoU        JBLK0100.BytAvl  *Zero
**
C          If         ApiRcvSiz
```

That loop keeps running the API until the results fit into the area of memory that was allocated. Each time through the loop, it calls the API, which returns the amount of memory that's needed to list all of the object locks in the JBLK0100.BytAvl field. If that field is larger than what's been allocated, the %realloc() BIF is called to increase the amount of allocated space until the area of memory is large enough to contain the entire result.

The JBLK0100 data structure only occupies the first 24 bytes of the memory that has been allocated to the pLstHdr pointer. The rest of the space will contain the list of objects that are locked. The API passes a field called OfsObjLck ("Offset to object locks") to tell the program where the list of object locks can be found.

An offset is a count of the number of bytes from a starting point to a desired spot in memory. In this case, the pLstHdr pointer is the starting point, so if the OfsObjLck value is added to the pLstHdr pointer, the result will be a pointer to the first entry in the list of object locks returned.

The following code will set the pLstEnt pointer to point to the start of the list of object locks:

```
C          Eval    pLstEnt = pLstHdr + JBLK0100.OfsObjLck
```

The JBLK0100E data structure is based in the area of memory pointed to by the pLstEnt pointer – that means that it now occupies the area of memory where the start of the list is stored! That means that the contents of the fields in JBLK0100E at this point in the code will be the first entry in the list.

The API passes back the LckObjEntLen field in the JBLK0100 data structure to tell us how large each element of the returned list is. In order to read the next element of the list, the JBLK0100E data structure has to be moved forward in memory by that amount. This is accomplished with the following code:

```
C          If      Eix
```

The IF statement ensures that the pointer doesn't get incremented an extra time on the last iteration of the FOR loop. Because of this, the pointer will never point to an area outside of the memory that's been allocated, which could potentially cause an error.

The Information Center page for the Retrieve Job Locks (QWCRJBLK) API can be found at the following link:
<http://publib.boulder.ibm.com/iserics/v5r2/ic2924/info/apis/qwcrjblk.htm>

You can download the sample code for this article from
<http://www2.systeminetwork.com/noderesources/code/clubtechcode/ListJobObjLocks.zip> .

The above source code was written by Carsten Flensburg. You can contact Carsten at <mailto:flensburg@novasol.dk> .

Source URL: <http://iprodeveloper.com/rpg-programming/apis-example-list-job-object-locks>