



Working with IFS Object Locks

[iPro Developer](#)

[Carsten Flensburg](#)

Carsten Flensburg

Tue, 08/28/2012 - 5:00am

APIs by Example

[Click here](#) to download the code bundle.
 To report code errors, email [iPro Developer](#)

When you need to locate the jobs currently holding locks against a specific object in the library file system QSYS.LIB, you can immediately do so using the Work with Object Locks (WRKOBJLCK) CL command. However, we presently have no similar command for objects in the IFS. Other options already exist, though. One is IBM Navigator for i, which includes a GUI that provides access to IFS objects and lock information. Another option is a virtually undocumented facility based on the Perform Miscellaneous File System Functions (QPoFPTOS) API that produces a printed list of IFS object lock holders. (For more information about these options, see the related references in "Find Out More" below.)

Sometimes, however, speed is of the essence, and running a CL command from a command line clearly outperforms the aforementioned alternatives. Because both options employ the Retrieve Object References (QPoLROR) API to retrieve object lock information, I made a shortcut to exploit this API to implement a Work with IFS Object Lock (WRKIFSLCK) CL command. Using the code I present in this article also gives you a starting point to create, for example, a Check IFS Object Lock (CHKIFSLCK) command to verify whether a specific IFS object is currently in use. You can use the code to write other similar useful CL commands to support a workflow or process involving the use of IFS objects as well.

QPoLROR's Required Parameters

The QPoLROR API documentation describes the API interface in C notation, as Figure 1 shows (for a list of IFS APIs, see the related link in "Find Out More").

Figure 1: QPoLROR API interface written in C

```
Retrieve Object References (QPoLROR) API

void QPoLROR(
    void *          Receiver_Ptr,
    unsigned int    Receiver_Length,
    char *          Format_Ptr,
    Qlg_Path_Name_T * Path_Ptr,
    void *          Error_Code_Ptr

);

Default Public Authority: *USE
```

Had IBM instead written the QPoLROR interface in the format commonly applied for most APIs in the Information Center API section, you'd have probably seen something along the lines of that in Figure 2.

Figure 2: QPoLROR API interface written in the common format

```
Retrieve Object References (QPoLROR) API

Required Parameter Group:

1 Receiver variable      Output      Char(*)
2 Receiver variable length Input      Unsigned binary(4)
3 Format name            Input      Char(8)
```

4	Path name	Input	Char (*)
5	Error code	I/O	Char (*)

Here, *Receiver variable* defines the storage location where the QPoLROR API places the IFS object lock information. The second parameter, *Receiver variable length*, defines the actual amount of space available for the information returned by the API, and *Format name* indicates the requested format of this information. As for the latter parameter, you have two options: format ROROo100 returns only the most crucial information and therefore requires less space than format ROROo200, which provides more comprehensive and detailed information. Let's first look at the ROROo100 format in Figure 3.

Figure 3: ROROo100 output format description (Qp0l_ROROo100_Output)

Header section with object level information plus offset and length of the following embedded structure:

```
Simple Object Reference Types Structure
(Qp0l_Sim_Ref_Types_Output)
```

When embedded in the ROROo100 format, the Simple Object Reference Types Structure will hold returned information that specifies general lock information for the IFS object defined in the QPoLROR API's *Path name* parameter. The Simple Object Reference Types Structure is also part of the more detailed lock information returned when you request the ROROo200 output format (Figure 4).

Figure 4: ROROo200 output format description (Qp0l_ROROo200_Output)

Header section with object level information plus offsets and lengths of the three following embedded structures:

```
Simple Object Reference Types Structure
(Qp0l_Sim_Ref_Types_Output)

Extended Object Reference Types Structure
(Qp0l_Ext_Ref_Types_Output)

Job Using Object Structure
(Qp0l_Job_Using_Object)

Simple Object Reference Types Structure
(Qp0l_Sim_Ref_Types_Output)

Extended Object Reference Types Structure
(Qp0l_Ext_Ref_Types_Output)

IBM i NetServer Session Using Object Structure
(Qp0l_Session_Using_Object Structure)
```

This structure listing shows the correlation between the different information structures, as well as the use of the Simple and Extended Object Reference Types structures at both the general IFS object level and the individual job level. In the former capacity, the lock information returned in the two structures represents the total number of known references defined by the various lock types for the IFS object. However, when we embed these structures within a specific job list entry, the returned lock information represents the number of references per lock type for the IFS object within that specific job.

Keeping in line with this explanation, the downloadable code accompanying this article includes the ROROo200 output format that has a header section pointing to three substructures, two of which provide IFS object lock information at two different detail levels. The third substructure—Job Using Object Structure—contains information pointing to yet another substructure layer, whose information pertains to each job currently holding one or more locks on the IFS object. The Job Using Object Structure is a repeating structure, meaning it has one entry for each returned job. To navigate the structure correctly, I use the structure's Displacement to Next Job Entry subfield as many times as the ROROo200 header structure's Jobs Returned subfield indicates, as

Figure 5 shows.

Figure 5: Using subfield Displacement to Next Job Entry

```

/Free

pJobUsgObj = %Addr( ROR00200 ) + ROR00200.OfsJobLst;

For   Idx = 1   to ROR00200.NbrJobRtn;

  If   JobUsgObj.DplSmpRef > *Zero;
    pSmpObjRef  = pJobUsgObj + JobUsgObj.DplSmpRef;
  EndIf;

  If   JobUsgObj.DplExtRef > *Zero;
    pExtObjRef  = pJobUsgObj + JobUsgObj.DplExtRef;
  EndIf;

  If   JobUsgObj.DplNetSvr > *Zero;
    pNetSvrRef  = pJobUsgObj + JobUsgObj.DplNetSvr;
  EndIf;

  //-- Process NetServer job list...

EndIf;

//-- Process job usage and object reference structures...

If   Idx < ROR00200.NbrJobRtn;
  pJobUsgObj += JobUsgObj.DplNxtJobE;
EndIf;
EndFor;

/End-Free

```

Back in Figure 2, the IFS object *Path name* parameter must be passed as a `Qlg_Path_Name_T` structure, which includes a path name or a pointer to a path name. (For more information about `Qlg_Path_Name_T`, see "APIs by Example: Zip and Unzip Files with the New 7.1 Zip API Support." To learn more about converting a path name from one Coded Character Set Identifier (CCSID) to another, see "APIs by Example: Conversion of a Path Name." You'll find both articles listed in "Find Out More.") To wrap up the discussion of the QPOLROR API parameter list, I've included the RPG IV prototype that implements the QPOLROR API interface (Figure 6).

Figure 6: RPG IV prototype implementing QPOLROR API interface

```

*--- Retrieve object references:
D  RtvObjRef      Pr               ExtPgm( 'QPOLROR' )
D  RcvVar          65535a          Options( *VarSize )
D  RcvVarLen       10u 0 Const
D  FmtNam          8a   Const
D  PthStr          5000a  Const  Options( *VarSize )
D  Error          32767a          Options( *VarSize )

```

WRKIFSLCK Command Prompt

Now it's time to look at the purpose of calling the QPOLROR API and extracting the returned IFS object lock information. Figure 7 displays the WRKIFSLCK command prompt, which accepts an IFS object path as its primary parameter.

Figure 7: Work with IFS Object Locks (WRKIFSLCK) command prompt

```

-----

                                Work with IFS Object Locks (WRKIFSLCK)

Type choices, press Enter.

IFS object . . . . .

```

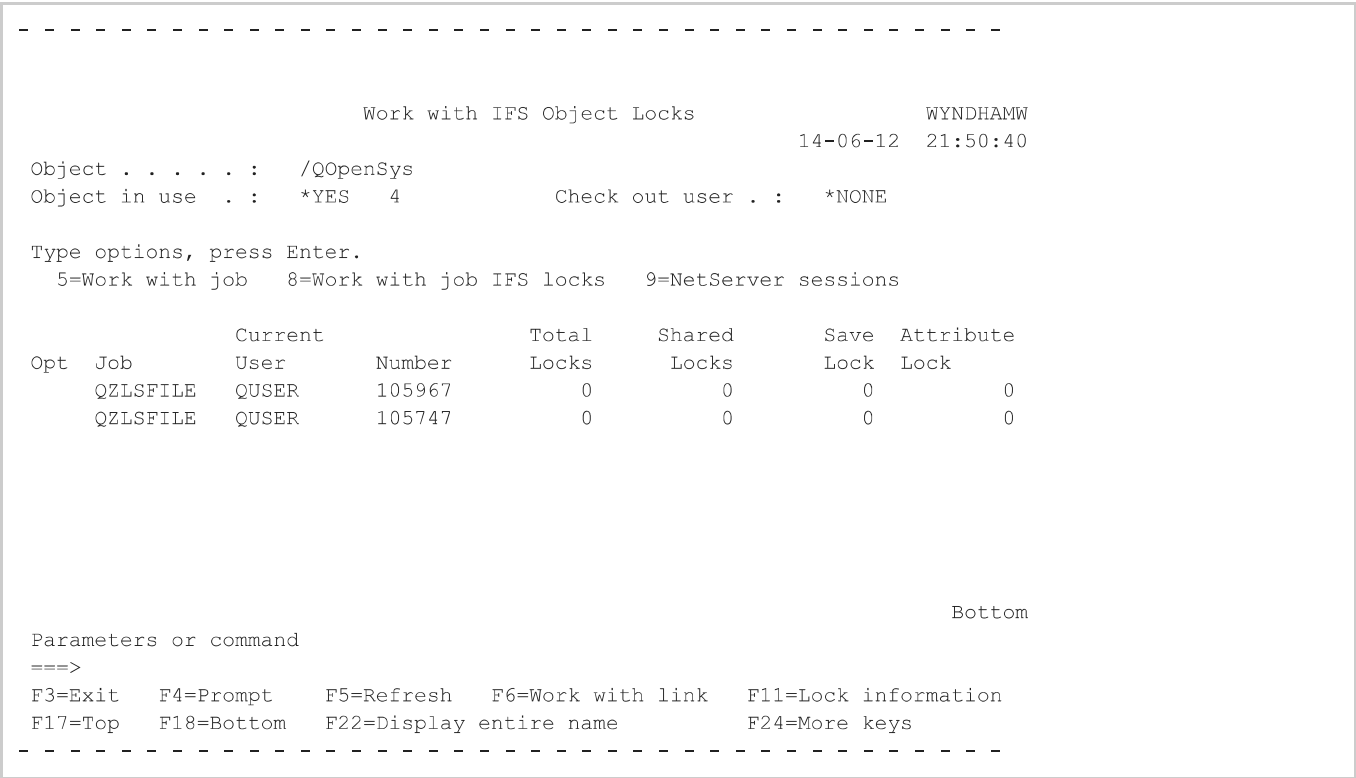
```
Output . . . . . * * *PRINT
-----
```

The command's second parameter defines whether the list of jobs currently holding one or more locks on the specified object should display in a list panel or print with your job's spooled output. To give you an example of the former output option, I ran the following command:

```
WRKIFSLCK OBJ('/QOpenSys')
```

Figure 8 shows the resulting list panel, which includes cursor-sensitive help text to explain the different parts of the panel as well as the list columns, list options, and function keys.

Figure 8: Work with IFS Object Locks list panel



Note that in an upcoming installment of APIs by Example, I'll provide a new CL command that will make available the *Work with job IFS locks* list option. Until that happens, this list option won't appear in the list panel. In case the IFS object name exceeds the panel space available, you can use the cursor position and F22 key to display the full IFS object name in a window.

How to Compile

Below, you'll find instructions on how to create the Work with IFS Object Locks command. The following sources are included with the code download associated with this article:

- CBX254—RPGLE: Work with IFS Object Locks - CPP
- CBX254E—RPGLE: Work with IFS Object Locks - UIM Exit Program
- CBX254—HPNLGRP: Work with IFS object Locks - Help
- CBX254P—PNLGRP: Work with IFS Object Locks - Panel Group
- CBX254V—RPGLE: Work with IFS Object Locks - VCP
- CBX254X—CMD: Work with IFS Object Locks

- CBX254M—CLP: Work with IFS Object Locks - Build command

To create all the above command objects, compile and run the CBX254M CL program, following the instructions in the source header. You'll also find compilation instructions in the respective source headers of the individual sources.

Find Out More

["Display File Usage Information"](#)

["Installation of Director 5.10.2 Fails on IBM i5/OS"](#) (this article shows an example of calling QPoFPTOS to list object locks)

["Renaming or Removing Files from the IFS That Have Invalid Names"](#) (IBM login required)

IBM I 7.1 Information Center documentation

[Allocating Resources](#)

[Integrated File System](#)

[Integrated File System APIs](#)

[Perform Miscellaneous File System Functions \(QPofPTOS\) API](#)

[Retrieve Object References \(QPofLROR\) API](#)

Articles at iProDeveloper.com

["APIs by Example: Zip and Unzip Files with the New 7.1 Zip API Support"](#)

["APIs by Example: Conversion of a Path Name"](#)

Source URL: <http://iprodeveloper.com/rpg-programming/working-ifs-object-locks>