

Working with Job IFS Object Locks

[iPro Developer](#)

[Carsten Flensburg](#)

Carsten Flensburg

Thu, 09/27/2012 - 3:00pm

APIs by Example

[Click here](#) to download the code bundle.

To report code errors, email [iPro Developer](#)

In "[APIs by Example: Working with IFS Object Locks.](#)" I discussed the options to identify jobs that are currently locking or accessing a specific object in the IFS. Likewise, you sometimes need to

determine which IFS objects a specific job is locking or accessing. For objects in the QSYS.LIB file system, there are a couple of ways to accomplish this: use either the Display Job (DSPJOB) or the Work with Job (WRKJOB) command and specify OPTION(*JOBLOCK) to display a list of QSYS.LIB objects currently locked by the job in question. To retrieve IFS object lock and access information, you can use the IBM Navigator for i GUI option or the Retrieve Referenced Objects (QPOLRRO) API.

In this article, I demonstrate how to code the QPOLRRO API and, to add a useful angle to that endeavor, present a Work with Job IFS Object Locks (WRKJOBIFSL) CL command based on QPOLRRO. (For a list of the source files and instructions for compiling this command, see the "How to Compile" box.) Using the WRKJOBIFSL together with the Work with IFS Object Locks (WRKIFSLCK) command included in the previously mentioned article will further equip you to investigate and handle IFS object locks. In doing so, you'll also learn how resource access in the IFS is controlled and IFS object integrity is enforced. (For GUI alternatives available for such research, see the articles "Display File Usage Information" and "Display Locks With Director Navigator" listed in the "Find Out More" box below.)

QPOLRRO's Required Parameters

As with the QPOLROR API that I discussed in last month's article, the QPOLRRO API documentation in the IBM Information Center presents the API interface in C notation. Transforming the specifications into a fully functional RPG IV prototype requires a little work from the RPG programmer. If at some point you encounter a similar challenge when trying to code an API, a great resource to help you overcome possible obstacles is the article "Converting from C Prototypes to RPG Prototypes" by IBM's Barbara Morris (just follow the link in the "Find Out More" box).

Figure 1 shows the original QPOLRRO API documentation.

Figure 1: Original QPOLRRO API documentation

Retrieve Referenced Objects (QPOLRRO) API

```
void QPOLRRO(
    void *          Receiver_Variable,
    unsigned int    Length_Of_Receiver_Variable,
    char *          Receiver_Format_Name,
    void *          Job_Identification,
    char *          Job_Identification_Format,
    void *          Error_Code
);
```

Default Public Authority: *USE

If you transform these specifications to the common API parameter interface format generally applied to the online API documentation, you'll probably end up with the format in Figure 2.

Figure 2: The QPOLRRO API's required parameters

Retrieve Referenced Objects (QP0LRRO) API

Required Parameter Group:

1	Receiver variable	Output	Char(*)
2	Receiver variable length	Input	Unsigned binary(4)
3	Format name	Input	Char(8)
4	Job identification information	Input	Char(*)
5	Job identification format	Input	Char(8)
6	Error code	I/O	Char(*)

Here, the *Receiver variable* parameter receives the IFS object lock information for the specified job returned by the QP0LRRO API. The *Receiver variable length* parameter defines to the API the size of the space available for the return information, and *Format name* specifies the requested format of this information. Currently, only the RROO0100 format is supported (Figure 3).

Figure 3: Format RROO0100

RROO0100 Output Format Description (Qp01_RROO0100_Output)

Header section with object list information plus offset to and length of the following object list header section:

Objects List Output Structure Description (Qp01_Obj_List_Output)

This structure is repeated for each IFS object referenced by the job and contains IFS object identifying attributes as well as the following displacement values and lengths:

- 1) Displacement to next Qp01_Obj_List_Output entry
- 2) Displacement to and length of the associated Qp01_Ext_Ref_Types_Output structure
- 3) Displacement to and length of the associated Qlg_Path_Name structure

The structures referenced at 2) and 3) immediately follow the list header section:

Extended Object Reference Types Structure
(Qp01_Ext_Ref_Types_Output)

Qlg path name structure identifying IFS object
(Qlg_Path_Name_T Structure)

In the command processing program for the WRKJOBIFSL command, all structures except RROO0100 are based on space pointers. Navigating the structures therefore involves setting these pointers to the address indicated by the preceding structure's displacement values. The displacement value defines the distance between the first byte of the current structure to the beginning of the substructure, so adding the displacement value to the current structure's address will map the based substructure to the correct storage location. Figure 4 shows the code used to extract all the IFS object lock information available in the RROO0100 return structure.

Figure 4: Extracting all IFS object lock information in the RROO0100 return structure

```

/Free

pQp01_Obj_List = %Addr( RROO0100 ) + RROO0100.OfsObjLst;

For Idx = 1 To RROO0100.ObjRtnCnt;

    pQp01_Ext_Ref_T = pQp01_Obj_List + Qp01_Obj_List.DplExtRefTyp;
    pQlg_Path_Name = pQp01_Obj_List + Qp01_Obj_List.DplPthNam;

    //-- Extract and convert path name - and process object lock information...

    If Idx < RROO0100.ObjRtnCnt;
        pQp01_Obj_List += Qp01_Obj_List.DplNxtObj;
    EndIf;
EndFor;

```

```
/End-Free
```

The fourth API parameter, *Job identification information*, identifies the job for which the IFS object lock information is returned. The two structures offered, JIDFo100 and JIDFo200, differ mainly in how they identify the job's appropriate thread level and how they scope the associated lock information. Both structures let you identify a job through a qualified job name, an internal job identifier, or the special value '*' pointing to the current job. The *Job identification format* parameter defines which of the two formats to use on the API call. The remaining API parameter—*Error code*—is the standard API error data structure. You can see in Figure 5 what the QPoLRRO API's parameter list amounts to when put together in an RPG IV prototype.

Figure 5: The QPoLRRO API's parameter list in an RPG IV prototype

```
***-- Retrieve job references:
D RtvJobRef      Pr              ExtPgm( 'QP0LRRO' )
D RcvVar          65535a        Options( *VarSize )
D RcvVarLen       10u 0 Const
D RcvFmt          8a  Const
D JobId           56a  Const
D JobIdFmt        8a  Const
D Error          32767a        Options( *VarSize )
```

The WRKJOBIFSL Command

Let's now turn to the WRKJOBIFSL command shown in Figure 6.

Figure 6: Work with Job IFS Object Locks (WRKJOBIFSL) command prompt

```
-----
                                Work with Job IFS Object Locks (WRKJOBIFSL)

Type choices, press Enter.

Job name . . . . . *              Name, *
User . . . . .      Name
Number . . . . .      000000-999999
Output . . . . . *      *, *PRINT
-----
```

The command accepts the qualified name of the job for which to list the currently locked IFS objects, as well as a second command parameter that defines whether the list of IFS objects should either display in a list panel or print with your job's spooled output. The following command will show in a work-with panel all IFS objects currently locked by the job running WRKJOBIFSL:

```
WRKJOBIFSL JOB(*) OUTPUT(*)
```

You can also select the WRKJOBIFSL command as a list option in the work-with list panel, which appeared in the WRKIFSLCK command presented last month.

Figure 7 shows an example of the initial list panel for WRKJOBIFSL. A total of four list views are available; the F11 key lets you toggle between the views.

Figure 7: Work with Job IFS Object Locks list panel

```
-----
                                Work with Job IFS Object Locks                                WYNDHAMW
                                                                                   16-07-12  10:55:40
-----
```

Job: WYNWPC01A User: CARSTEN Number: 229600

Type options, press Enter.

5=Work with link 8=Work with IFS object locks

Opt	Object	Total Locks	Shared Locks	In use	Reference Count
/		0	0	*YES	2
/QOpenSys		2	1	*YES	1

Bottom

Parameters or command

==>

F3=Exit F4=Prompt F5=Refresh F6=Work with job F11=Lock information
F17=Top F18=Bottom F22=Display entire name F24=More keys

As always, the command includes cursor-sensitive help text to explain the different parts of the panel as well as the list columns, list options, and function keys. If the IFS object name exceeds the list column space available, you can use the cursor position to identify the IFS object and press the F22 key to display the full IFS object name in a window.

If you're interested in a discussion about and code examples of object and record lock APIs targeting library objects in QSYS.LIB, you can find links to this topic in the "Find Out More" box.

How to Compile

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

- CBX255—RPGLE: Work with Job IFS Object Locks-CPP
- CBX255E—RPGLE: Work with Job IFS Object Locks-UIM Exit Program
- CBX255H—PNLGRP: Work with Job IFS Object Locks
- CBX255P—PNLGRP: Work with Job IFS Object Locks-Panel Group
- CBX255V—RPGLE: Work with Job IFS Object Locks-VCP
- CBX255X—CMD: Work with Job IFS Object Locks
- CBX255M—CLP: Work with Job IFS Object Locks-Build command

To create all of these command objects, compile and run the CBX255M CL program, and follow the instructions in the source header. You'll also find compilation instructions in the respective source headers of the individual sources.

Find Out More

["Converting from C prototypes to RPG prototypes"](#)

["Display File Usage Information"](#)

["Display Locks With Director Navigator"](#)

IBM i 7.1 Information Center documentation

[Integrated File System](#)

[Integrated File System APIs](#)

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

[Retrieve Referenced Objects \(OPoLRRO\) API](#)

Articles at iProDeveloper.com

["Easily Respond to Object and Record Lock Events"](#)

["Process File Lock \(PrcFilLck\) Utility"](#)

["Process Record Locks \(PrcRcdLck\) Utility"](#)

["Working with IFS Object Locks"](#)

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