



APIs by Example: Working with Printers -- All Printers, Please!

[Carsten Flensburg](#)

Tue, 02/25/2014 - 12:02pm

Manage local, LAN, and remote output queue writers with one command

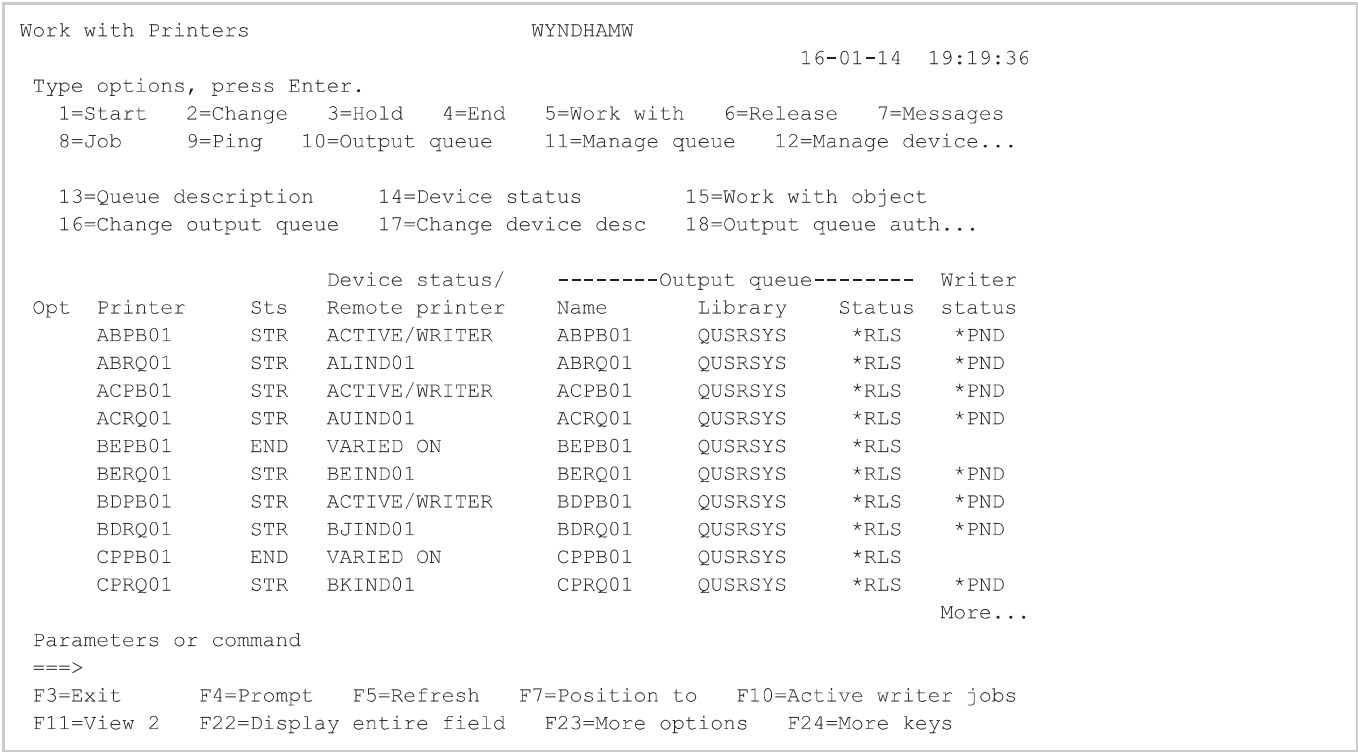
In the previous installment of APIs by Example, “[Work Management APIs: Keeping an Eye on Job Activity](#),” I noted that the beauty of APIs is that they give you the opportunity to accommodate specific requirements as challenges arise. The *Work with Printers* (WRKPRT) command we’ll examine in this article is yet another example that demonstrates the power and versatility of API programming. To cover the requirements part of the equation for this article, I was fortunate to receive help and inspiration from Peter Kemp of Melbourne, Australia.

I’ve previously published the *Work with LAN Printers* (WRKLANPRT) and *Work with Remote Output Queues* (WRKRMTOUQ) commands, which provide management capabilities to LAN printers and remote output queue writers, respectively. Peter suggested a WRKPRT command that would cover local, LAN, and remote output queue writers, all in one. He also recommended that I add an *OUTFILE output option to this command, similar to the feature in Bryan Dietz’s excellent *Work with Remote Writer* (WRKRMWTR) utility available at Bryan’s [IBM i freeware page](#). The outcome of the design, build, and test efforts that ensued is the topic of this article. To learn how to create the WRKJOBS command, see the “How to Compile” section below.

Designing WRKPRT

The basic design objectives for WRKPRT included a *Work with Printers* list panel (Figure 1) that would present the most basic and operational attributes of the printers and remote output queues involved.

Figure 1: Work with Printers list panel



The purpose was to enable an overview of the printer configuration and current operational status. In addition to displaying printer status, the list panel shows other details for the device status/remote output queue, output queue name and status, writer status, remote address/system, message queue, and remote output queue attributes controlling the printer data stream. WRKPRT contains support for multiple writers that are started against the same output queue, as well as a position list function key for fast navigation of the list.

In addition, no less than 18 list options are available in the list panel, giving you options to quickly manage all aspects of printers, writer jobs, and output queues:

- 1=Start
- 2=Change
- 3=Hold
- 4=End
- 5=Work with
- 6=Release
- 7=Messages
- 8=Job
- 9=Ping
- 10=Output queue
- 11=Manage queue
- 12=Manage device
- 13=Queue description
- 14=Device status
- 15=Work with object
- 16=Change output queue
- 17=Change device desc
- 18=Output queue auth

The *F23=More options* key lets you conditionally display all list options. You can also direct the output from the WRKPRT command to either a spooled file or a database output file. In addition, several other function keys provide shortcuts to the native printer management commands.

The List Configuration Descriptions (QDCLCFGD) API and the List Objects (QUSLOBJ) API deliver the core functionality of the WRKPRT command processing program, and both APIs return the produced information to user spaces. The QDCLCFGD API can list all types of configuration object information, including printer devices in general, but it also lets you narrow the returned information to local or LAN printer devices specifically. The QUSLOBJ API returns information about all types of objects, and in this context, it lists objects of type *Output queue* (*OUTQ) found in the library specified in the WRKPRT command prompt (Figure 2).

Figure 2: Work with Printers (WRKPRT) command prompt

```
Work with Printers (WRKPRT)

Type choices, press Enter.

Printer name . . . . . *ALL      Name, generic*, *ALL
Type . . . . . *ALL      *ALL, *LCL, *LAN, *OUTQ
Status . . . . . *ALL      *ALL, *STR, *END, *HLD...
Output . . . . . *        *, *PRINT, *OUTFILE
Remote output queue library . . *ALL  Name, *LIBL, *CURLIB, *ALL
Output file . . . . .      Name
Library . . . . . *LIBL    Name, *LIBL, *CURLIB
Replace or add records . . . . . *ADD  *ADD, *REPLACE
```

The QUSLOBJ API supports all special values for the indicated library, so you can scope the returned list to your preferred extent—including all libraries available on the system—thus exceeding the qualification of the job’s current library list. For output queues, you have the option of using the more recent Open List of Output Queues (OSPOLOTQ) API, which specifically targets listing of output queues. (An IBM i 6.1 PTF introduced the QSPOLOTQ API, and it now comes with the base operating system at 7.1.) Handling output from open list APIs is conceptually and programmatically different from managing user space-based APIs. For details about the open list APIs, see “[APIs at Work—with Jobs.](#)”

To reduce the complexity of the code, I took the simple approach of employing user space-based APIs only. As Figure 3 shows, the List Configuration Descriptions (QDCLCFGD) API parameter list is straightforward.

Figure 3: List Configuration Descriptions (QDCLCFGD) API parameters

```
List Configuration Descriptions (QDCLCFGD) API
```

Required Parameter Group:

1	Qualified user space name	Input	Char (20)
2	Format name	Input	Char (8)
3	Configuration description type	Input	Char (10)
4	Object qualifier	Input	Char (40)
5	Status qualifier	Input	Char (20)
6	Error code	I/O	Char (*)

Default Public Authority: *USE

In consecutive order, you tell the API the qualified name of the user space that will receive the returned information, the format name defining the layout of this information, and the configuration object type (e.g., *CTLD, *LIND, or in this case, *DEVDD). Next, you further qualify the configuration objects to return, either by group (such as *PRT, meaning all devices that are printers) or by type (such as *LANPRT, meaning only LAN printers). Finally, you specify a status qualifier to narrow the list to only the configuration objects that comply with the specified status, succeeded by the common API error data structure. (For more details on the object and status qualifier parameters, please refer to the API manual.)

Although the List Objects (QUSLOBJ) API parameter list in Figure 4 supports up to eight parameters, only four are required.

Figure 4: List Objects (QUSLOBJ) API parameters

List Objects (QUSLOBJ) API

Required Parameter Group:

1	Qualified user space object	Input	Char (20)
2	Format name	Input	Char (8)
3	Object and library name	Input	Char (20)
4	Object type	Input	Char (10)

Optional Parameter Group 1:

5	Error Code	I/O	Char (*)
---	------------	-----	----------

Optional Parameter Group 2:

6	Authority control	Input	Char (*)
7	Selection control	Input	Char (*)

Optional Parameter Group 3:

8	Auxiliary storage pool (ASP) control	Input	Char (*)
---	--------------------------------------	-------	----------

Default Public Authority: *USE

In this example, I employ five of these parameters. As with the QDCLCFGD API, you first specify the qualified name of the user space to receive the return information, followed by the format name specifying its layout. The third and fourth parameters define the qualified object name and the object type, respectively. Special values are supported for both parameters. The final (and optional) parameter I use is the API error data structure. For more information about user space APIs and how to retrieve and process user space content, see [“APIs by Example: Retrieve Subsystem Entries API.”](#)

Keeping the WRKPRT Command Tidy

Housekeeping is an important aspect of including user spaces in your design. You'll typically need to create the user space as part of your initiation routines in the program employing the user space. Likewise, it's good practice to delete the user space when you terminate your program to release all allocated resources. Careful consideration is therefore required when naming your user space; simply assigning a specific name might get you into trouble if your program is called recursively, as would be the case if you ran the WRKPRT command from the list panel's command line.

When the newest invocation of the command processing program terminates, it will delete the user space, and

on return to the initial invocation of the program, that program will run into an exception on the next attempt to access the previously deleted user space. You can handle the situation in several different ways, mainly either by naming the user space uniquely or by verifying the existence of the user space before each attempted access, and then creating the user space if necessary. The option I usually choose is to ensure a unique name by appending the current call level to a six-character name.

The [Retrieve Call Stack \(OWVRCSTK\) API](#) will return the current number of active call levels in the header information of the return information data structure. I've wrapped all the code generating the four digits in the procedure GetInvLvl() in the command processing program CBX266 (I suggest you look this up for the details). I then appended the four leftmost digits returned by GetInvLvl() to the fixed portion of the user space name and thus constructed a user space name that's unique within the current job, as this code snippet shows:

```
UsrSpc_q = USRSPC + GetInvLvl() + 'QTEMP';
```

Bonus Commands and Exit Program

My thanks to Peter Kemp for his assistance in conceiving, creating, and testing the WRKPRT command. On a side note, Peter played an equally important role in creating six new job schedule entry commands published earlier in this article series. To learn more about these new job schedule entry commands, see the related articles in the "Find Out More" section.

I also want to mention that for added capabilities such as supporting the number of copies attribute of a spooled file to remote output queues, you can download the [TSPRWPR exit program](#), written by IBM's Rodney Johnson, former technical lead of System i Print. In the IBM Technical Documentation section of "Find Out More" you'll find links to additional information on the configuration of remote output queues and LAN printers.

How to Compile

These instructions show you how to create the Work with Jobs (WRKJOBS) command and all its associated commands and objects. The following sources are included with the code download available with this article:

CBX209—RPGLE: Additional Message Information

CBX209E—RPGLE: Additional Message Information—UIM Exit Program

CBX209H—PNLGRP: Additional Message Information—Help

CBX209P—PNLGRP: Additional Message Information—Panel Group

CBX971—RPGLE: Work with Output Queue Authorities—CPP

CBX971E—RPGLE: Work with Output Queue Authorities—UIM Exit Pgm

CBX971H—PNLGRP: Work with Output Queue Authorities—Help

CBX971P—PNLGRP: Work with Output Queue Authorities—Panel Group

CBX971V—RPGLE: Work with Output Queue Authorities—VCP

CBX971X—CMD: Work with Output Queue Authorities

CBX971M—CLP: Work with Output Queue Authorities—Build command

CBX266—RPGLE: Work with Printers—CPP

CBX266E—RPGLE: Work with Printers—UIM Exit Program

CBX266H—PNLGRP: Work with Printers—Help

CBX266P—PNLGRP: Work with Printers—Panel Group

CBX266V—RPGLE: Work with Printers—VCP

CBX266X—CMD: Work with Printers

CBX266M—CLP: Work with Printers—Build Command

To create these commands and associated objects, compile and run the CBX266M CL program and all the other CL programs in the above list, following the instructions in the source header. You'll also find compilation instructions in the respective source headers of the individual sources. Note that for the WRKPRT command processing program CBX266 to compile, you must download a copy of the SQLCLI_H member (created by Scott Klement) to a QRPGLSRC source file in your job's library list. I've provided a link to a zip file containing the correct version of the SQLCLI_H copy member in the "Find Out More" section.

Find Out More

[“APIs at Work—with Jobs”](#)

[“APIs by Example: Directing API Output to Output Files Using the SQL CLI APIs”](#)

[“APIs by Example: Message Handling APIs & Additional Message Info Support”](#)

[“APIs by Example: Retrieve Subsystem Entries API”](#)

[“APIs by Example: Work with LAN Printers Command”](#)

[“Are Your Sensitive Reports Secured? New Command WRKOUTQAUT”](#)

[“Carsten's Corner - New Work with Remote Output Queue Command”](#)

[SQLCLI_H copy member](#) (zip-file)

[“Work Management APIs: Keeping an Eye on Job Activity”](#)

[“Work Management APIs: Putting the Pieces Together”](#)

[Work with Remote Writer](#) (Bryan Dietz)

[“APIs by Example: One Job Schedule Entry API and Four New Job Schedule Entry Commands”](#)

[“Carsten's Corner: List Job Schedule Entries to an Output File”](#)

[“Carsten's Corner: Merge, Migrate, or Copy Job Schedule Entries Between Systems”](#)

IBM i 7.1 Information Center Documentation

[List Configuration Descriptions \(QDCLCFGD\) API](#)

[List Objects \(QUSLOBJ\) API](#)

[Retrieve Call Stack \(QWVRCSTK\) API](#)

[Retrieve Database File Description \(QDBRTVFD\) API](#)

[Retrieve Device Description \(QDCRDEVD\) API](#)

[Retrieve Output Queue Information \(QSPROUTQ\) API](#)

[Retrieve Writer Information \(QSPRWTRI\) API](#)

IBM Technical Documents

[Capabilities and Limitations of *LAN 3812 Printer Device Descriptions](#)

[Capabilities and Limitations of Remote Output Queues \(RMTOUTOs\)](#)

[Configuration Settings and Error Messages for *LAN 3812 PJJ Device Descriptions](#)

[Configuration Settings and Error Messages for *LAN 3812 SNMP Device Descriptions](#)

[Configuration Settings and Error Messages for Remote Output Queues \(RMTOUTOs\)](#)

[Configuring a *LAN 3812 Device Description that Uses the LPR Print Driver \(TSPLPRD\) Exit Program](#)

[Configuring a *LAN 3812 PJJ Device Description](#)

[Configuring a *LAN 3812 SNMP Device Description](#)

[Configuring a Remote Output Queue \(RMTOUTQ\)](#)

[Options to Configure a Printer on an IBM System i](#)

[TSPRWPR Remote Writer Page Range Support Exit Program](#)

Content Classification: Influencer

Source URL: <http://iprodeveloper.com/application-development/apis-example-working-printers-all-printers-please>