

APIs by Example: Keeping Track of Your Exports

[iPro Developer](#)
[Carsten Flensburg](#)

Thu, 02/21/2013 - 6:00am

Find modules that export procedures with the WRKPGMEXP command

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

Sometimes, it's handy to be able to quickly locate the module or service program that exports a specific procedure, or a specific data item, for that matter. You might, for example, need a particular routine you wrote a while ago but no longer recall in which service program it's located. Or perhaps you want to write a new procedure but must first ensure that a procedure by that name doesn't already exist. Because I've often encountered such situations, I decided to write the *Work with Program Export* (WRKPGMEXP) command. (For instructions on how to create WRKPGMEXP, see the "How to Compile" box below.)

How WRKPGMEXP Works

The WRKPGMEXP command lists all programs and service programs containing modules that export a specified symbol name, which defines either a procedure or data item. For WRKPGMEXP to retrieve the exported symbol names, the modules bound into the programs or service programs must still exist as objects in the library from which they were bound. By using binder language, you can prevent the export of a symbol from a service program, even if the symbol is exported from the module used to create that service program. Therefore, the command supports an option that lets you retrieve symbol names from the service program instead.

A program object doesn't list the procedures and data items exported from the modules bound into the program because the program itself doesn't export symbols, but a service program does. A service program exports all symbols defined with the EXPORT keyword in the modules indicated in the Create Service Program (CRTSRVPGM) command if you stipulate *ALL for the command's export (EXPORT) parameter. Alternatively, specifying EXPORT(*SRCFILE) will export only those symbols declared in the binder language and defined in the source member indicated for the CRTSRVPGM command's export source file (SRCFILE) and export source member (SRCMBR) parameters.

The Required Parameters

To perform the heavy-duty tasks of the WRKPGMEXP command, I use three Program and CL Command APIs: List Service Program Information (QBNLSPGM), List Module Information (QBNLMODI), and List ILE Program Information (QBNLPGMI). All three APIs have the same interface, which displays the four required parameters in Figure 1.

1	Qualified user space name	Input	Char(20)
2	Format name	Input	Char(8)
3	Qualified object name	Input	Char(20)
4	Error Code	I/O	Char(*)

Default Public Authority: *USE

Figure 1: The APIs' required parameter group

The *Qualified user space name* parameter defines the user space where the API will return the requested module, program, or service program information. The *Format name* parameter defines the format of the returned data. Figure 2 shows the types of information that the three APIs currently support.

Module		Program		Service program	
Format:	Detail:	Format:	Detail:	Format:	Detail:
MODL0100	*EXPORT				
MODL0200	*IMPORT				

	PGML0100	*MODULE	SPGL0100	*MODULE
	PGML0110	*MODULE extended	SPGL0110	*MODULE extended
	PGML0200	*SRVPGM	SPGL0200	*SRVPGM
	PGML0300	*ACTGRPEXP	SPGL0300	*ACTGRPEXP
	PGML0400	*ACTGRPIMP	SPGL0400	*ACTGRPIMP
MODL0300	*PROCLIST			
MODL0400	*REFSYSOBJ			
MODL0500	*COPYRIGHT	PGML0500	*COPYRIGHT	
			SPGL0600	*PROCEXP
			SPGL0610	*PROCEXP extended
			SPGL0700	*DTAEXP
			SPGL0800	*SIGNATURE

Figure 2: Formats defined for module, program, and service program

As you can see, the corresponding Display Module (DSPMOD), Display Program (DSPPGM), and Display Service Program (DSPSRVPGM) commands' detail (DETAIL) option defines each format. For more about the various types of information, look up the API documentation or the DETAIL parameter's help text for each of the CL commands.

The third API parameter in Figure 1—*Qualified object name*—defines the object in question as a module, program, or service program name. The final API parameter, *Error Code*, is the standard API error data structure.

Creating the *Work-With* List

To build the program and service program *work-with* list, follow these steps:

- 1. Call the Open List of Objects (QGYOLOBJ) API to list all programs and service programs identified by the WRKPGMEXP command's REFPGM parameter.
- 2. For each program found, list all modules bound into the program or service program using the QBNLPGMI API. For each service program found, use the QBNLSPGM API, and:
 - a. If you specified the service program option SRVPGMOPT(*MODULE), list the modules bound into the service program.
 - b. If you used SRVPGMOPT(*SRVPGM), list the service program exports and match each export symbol name and type against the symbol name and type indicated as the WRKPGMEXP command's EXPORT parameter.
- 3. Perform the following verification and investigation process for each bound module:
 - a. Call the Retrieve Object Description (QUSROBJD) API to check whether the module object exists.
 - b. If the module object does exist, use the Retrieve Module Information (QBNRMODI) API to verify the module source level against the corresponding data recorded into the program object.
 - c. If step 3b passes verification, use the QBNLMODI API to retrieve and process the module export symbol list, and match each export symbol name and type against the symbol name and type specified as the WRKPGMEXP command's EXPORT parameter.
- 4. If step 3c produces a match, the WRKPGMEXP command's *work-with* list will provide program and module information. Alternatively, if step 2b runs and produces a match, the *work-with* list will include service program information.

Note that if you specify the value *VFYMODREF for the WRKPGMEXP command's EXPORT parameter, the *work-with* list will include only modules that failed either test in step 2a or 2b and will ignore the SRVPGMOPT parameter. Figure 3 shows the WRKPGMEXP command's prompt panel.

```
-----
                                Work with Program Export (WRKPGMEXP)

Type choices, press Enter.

Exported symbol name . . . . .
Export symbol type . . . . .      *PROC          *PROC, *DATAITEM
Export program . . . . .          Name, generic*, *ALL
Library . . . . .                 *LIBL          Name, *LIBL, *CURLIB...
Export program type . . . . .     *ANY           *ANY, *PGM, *SRVPGM
Service program option . . . . .  *MODULE       *MODULE, *SRVPGM
```

```
Sort order . . . . . *OBJLIB      *OBJLIB, *TYPEOBJ, *LIBOBJ...
Output . . . . . *              *, *PRINT
-----
```

Figure 3: Work-with Program Export (WRKPGMEXP) command prompt

Here, you specify as the primary parameter the name of the procedure or data item for which you want to find all program references—and stipulate the symbol type as the secondary parameter. Please note that the symbol name is case-sensitive, as are export symbol names. Next, enter the generic name and library qualification of the programs and service programs whose modules you want to list and check for unresolved exports. Using the special name value **ALL* and one of the special values available for library qualification, you can potentially list and examine numerous programs and service programs. This technique can lead to extensive use of system resources, so you should narrow the selection range as much as possible.

You also have the option of specifying only one program type to include in the array of programs to examine, as well as how service programs are processed; the symbol list is resolved either from the modules bound into the service program or from the service program itself. As mentioned earlier, a different outcome might result if you use binder language to create the service program export list, or if the modules bound into the service program no longer exist. Alternatively, you can specify a variety of sort orders for the produced list. Finally, you can choose to print the program list instead of displaying a *work-with* panel. The online help text offers more details about the command and its parameters.

When I run the following command on my system:

```
WRKPGMEXP EXPORT(GETCTRCOD)
          SYMTYP(*PROC)
          EXPPGM(QGPL/WEB*)
          EXPPGMTYP(*ANY)
          SRVPGMOPT(*MODULE)
          ORDER(*OBJLIB)
          OUTPUT(*)
```

I'm presented with the *work-with* panel shown in Figure 4.

```
-----
                                Work with Program Export                                WYNDHAMW
                                                                29-11-12  11:20:21

Export symbol . . . . . :  GETCTRCOD
Symbol type . . . . . :   *PROC

Type options, press Enter.
 2=Update          4=Delete program   5=Display program   6=Display module
 7=Work with PDM   8=Program reference 9=Module reference

  Program          Library    Type      Module          Library    Module
Opt  Name          Library    Type      Name          Library    Status
-----
WEB421  QGPL        *PGM      WEB421  QGPL        *EXPSYMFND
WEB422  QGPL        *PGM      WEB421  QGPL        *EXPSYMFND
WEB423  QGPL        *PGM      WEB421  QGPL        *EXPSYMFND
WEB530  QGPL        *PGM      WEB530  QGPL        *EXPSYMFND
WEB543  QGPL        *PGM      WEB643  QGPL        *EXPSYMFND
WEB650  QGPL        *PGM      WEB650  QGPL        *EXPSYMFND

                                                                More...

Parameters or command
===>
F3=Exit      F4=Prompt   F5=Refresh   F9=Retrieve   F11=Display program text
F12=Cancel   F17=Top      F18=Bottom   F22=Display entire export symbol
-----
```

Figure 4: Work-with Program Export list panel

Here, the F11 key lets you toggle between the various types of program and module information exposed by the list panel. Using the list options, you can immediately execute an array of program and service program

commands against the found objects. The list of commands includes UPDPGM/UPDSRVPGM, DLTPGM/DLTSRVPGM, DSPPGM/DSPSRVPGM, DSPMOD, WRKMBRPDM, and DSPPGMREF. As always, online help text that explains the list panel, columns, options, and function keys is available.

To check the result of using the WRKPGMEXP command's SRVPGMOPT(*SRVPGM) option, you can run the following command:

```
WRKPGMEXP EXPORT ('_EXCP_MSGID')  
          SYMTYP (*DATAITEM)  
          EXPPGM (QSYS/*ALL)  
          EXPPGMTYP (*SRVPGM)  
          SRVPGMOPT (*SRVPGM)
```

Afterward, you should see a service program named QC2UTIL1 in library QSYS. If the QC2UTIL1 service program has been replaced by one or more PTFs on your system, you might also locate service programs with names beginning with QPZA*.

In previous APIs by Example articles, I've published utilities that let you locate programs or service programs that import a specific procedure or data item, as well as programs or service programs that reference a particular service program. You'll find links to the articles presenting the *Work with Program Import* (WRKPGMIMP) and the *Work with Service Program References* (WRKSPGREF) commands in the "Find Out More" box below.

How to Compile

Below are instructions on how to create the *Work with Program Export* command. The code download associated with this article includes the following sources:

CBX259—RPGLE: Work with Program Export—CPP

CBX259E—RPGLE: Work with Program Export—UIM Exit

CBX259H—PNLGRP: Work with Program Export—Help

CBX259P—PNLGRP: Work with Program Export—Panel Group

CBX259V—RPGLE: Work with Program Export—VCP

CBX259X—CMD: Work with Program Export

CBX259M—CLP: Work with Program Export—Build Command

To create all of these command objects, compile and run the CBX259M 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

[IBM i ILE Concepts 7.1](#)

[Moving to Integrated Language Environment for RPG IV](#)

IBM i 7.1 Information Center documentation

[List ILE Program Information \(OBNLPGMI\) API](#)

[List Module Information \(OBNLMODI\) API](#)

[List Service Program Information \(OBNLSPGM\) API](#)

[Retrieve Module Information \(OBNRMODI\) API](#)

[Retrieve Program Information \(OCLRPGMI\) API](#)

Articles at iProDeveloper.com

["APIs by Example: Identifying and Working with Service Program References"](#)

["APIs by Example: Locating and Working with Module Imports"](#)

["Binder Language and the Signature Debate"](#)

["Meat of the Matter: In RPG, Subprocedures Are Useful"](#)

["RPG Name Spaces"—forum discussion](#)

Source URL: <http://iprodeveloper.com/application-development/apis-example-keeping-track-your-exports>