

[print](#) | [close](#)

APIs by Example: Release 6.1 Change Program Data API PTF'ed to 5.4

[*System iNetwork Programming Tips Newsletter*](#)

[Carsten Flensburg](#)

Carsten Flensburg

Thu, 12/10/2009 (All day)

The Change Program Data (QBNCHGPD) API introduced with release 6.1 has also been made available for release 5.4 by means of a couple of PTFs. Although IBM released this information in a Software Knowledge Base Document published early this year, it did not come to my attention until mentioned by list owner and moderator David Gibbs in a thread on midrange.com recently. The thread took its origin in a requirement of being able to source debug a program module compiled with debug view (DBGVIEW) *SOURCE, where the source file location being pointed to in the module object had changed since the module creation.

Another use of the QBNCHGPD API would, for example, be where bound modules had been moved to other libraries following the creation of the programs or service programs into which they are bound, yet forever pointing to the original module library and thus making the tracking and verification of this relationship correspondingly difficult. The QBNCHGPD API allows you to change the data stored in the module, program, or service program object pointing to these references, without having to re-create the object from scratch. In fact, the following type of information can be modified: Object creation source file, service program export source file, bound module library, and bound service program library. More details on this in a minute. The API supports both OPM and ILE programs as well as modules and service programs.

In order to precisely target the exact program data to be changed among the potentially numerous objects referenced within a single program or service program, the QBNCHGPD API offers an array of input parameters. At the global level, you have the option of specifying a bound module name as well as a bound service program name parameter, both of which accept the special value *ALL, a generic name, or a specific name. Only the modules and service program references matching the specified criteria are updated. Plus, for each of the program data types, eligible to be changed, further selection criteria are available, as I show you shortly. But first a list of the program data types that can be changed by the QBNCHGPD API:

Creation source files

The Creation source files stored in the selected objects. This information can be changed for all object types. For ILE program and service program objects, the creation source file fields apply to the bound modules.

Export source file

The Export source file stored in the selected objects. This information can be changed for service programs only.

Bound module library

The Bound module library stored in the selected objects. This information can be changed for ILE programs and service programs only.

Bound service program library

The Bound service program library stored in the selected objects. This information can be changed for ILE programs and service programs only.

To help me explain the selection criteria designated for each individual type of program data, let me begin with a look at the CHGPGMDTA command's prompt panel:

Change Program Data (CHGPGMDTA)		
Type choices, press Enter.		
Object		Name
Library	*LIBL	Name, *LIBL,
*CURLIB		
Object type	*PGM	*MODULE, *PGM,
*SRVPGM		
Bound module name	*ALL	Name, generic*,
*NONE, *ALL		
Library	*ALL	Name, generic*,
*ALL		
Bound service program name . . .	*ALL	Name, generic*,
*NONE, *ALL		
Library	*ALL	Name, generic*,
*ALL		
Change parameters:		
Program data	*CRTSRCFIL	*CRTSRCFIL,
*EXPSRCFIL...		
From library	*ALL	Name, generic*,
*ALL		
From file	*ALL	Name, generic*,
*ALL, *BLANK		
From member	*ALL	Name, generic*,
*ALL, *BLANK		
To library	*SAME	Name, *SAME
To file	*SAME	Name, *SAME, *BLANK
To member	*SAME	Name, *SAME, *BLANK
+ for more values		

You can specify all four program data types on a single CHGPGMDTA command call, each allowing you to further pinpoint the program data to be changed by means of three *From* parameters, *Library*, *File*, and *Member name*, respectively. Each of the three parameters supports the special value *ALL, a generic name, as well as a specific name. These parameters imply that only program

data currently containing values matching the ones specified are updated with the values entered for the corresponding three *To* parameters. Adding to these parameters the mentioned global bound global module and service program name and you're able to target quite flexibly and precisely the program data to be changed. I show you examples demonstrating the use and interaction of the selection parameters below.

Constraints apply to the various program data types. In general, integrity checks are performed for the objects and source members specified for the *To* parameters, and if the check fails, so will the Change Program Data command and API. Specifying a *From member* value that is either a generic name or the special value *ALL in conjunction with a specific *To member* name is for what seems like quite logical reasons also disallowed. Further, if you want to change a bound module's or bound service program's library name, you must specify the special value *BLANK for the *From file* and *From member* as well as the *To file* and *To member* parameters in order for the command and API to run successfully. Finally, observable creation data must also be available in programs being changed. In case of the CHGPGMDTA command failing to complete normally, an escape message is returned to the caller, providing information about the cause of failure.

The 5.4 version of the QBNCHGPD API offers a quite simplistic version of the regular API error-reporting facility. Instead of returning individual message IDs for the respective types of errors encountered when calling the API, the general application escape message identifier CPF9898 is returned and the cause of the error spelled out (in all capital letters) in the message text provided as the message's message data. I haven't had a chance to verify on a release 6.1 system, but if the 6.1 API documentation is considered a reliable source of information, the regular API error-reporting convention has been reinstated at this release level. You can find links to the release 5.4 cover letters for the PTFs making the QBNCHGPD API available for this release. If at this release, you must load and apply the mentioned PTFs prior to running the CHGPGMDTA command. For earlier releases the QBNCHGPD API is not available.

For now, let us have a look at some examples of what the CHGPGMDTA command (and API) is able to accomplish. The following command changes the source file library reference of the specified module from library OLDLIB to library NEWLIB:

```
CHGPGMDTA OBJ(NEWLIB/CBX210)
          OBJTYPE(*MODULE)
          CHGPARAM(*CRTSRCFIL OLDLIB *ALL *ALL NEWLIB *SAME *SAME))
```

To change the library of a program's module reference you would run the following command:

```
CHGPGMDTA OBJ(NEWLIB/CBX210)
          OBJTYPE(*PGM)
          BNDMODULE(OLDLIB/CBX210)
          CHGPARAM(*BNDMODLIB OLDLIB *BLANK *BLANK NEWLIB *BLANK
*BLANK) )
```

This would cause the library specified for the module in the program's module list located on the Display Program (DSPPGM) command's third page (DSPPGM DETAIL(*MODULE)) to be changed from the name OLDLIB to the name NEWLIB. If the module in question also contained the program entry procedure for the program, this reference displayed on the DSPPGM command's first page (DSPPGM DETAIL(*BASIC)) would reflect the change as well.

Running the command below will change the source file library name for the bound module reference CBX210 in the program of the same name from OLDLIB to NEWLIB. The information changed is the one found in a program's module list located on the Display Program (DSPPGM) command's third page (DSPPGM DETAIL(*MODULE)), when specifying option 5 for the module in question.

```
CHGPGMDTA OBJ(NEWLIB/CBX210)
          OBJTYPE(*PGM)
          BNDMODULE(NEWLIB/CBX210)
          CHGPARM(*CRTSRCFIL OLDLIB QRPGLSRC CBX210 NEWLIB
QRPGLSRC CBX210))
```

To change the library value *LIBL to NEWLIB for service program CBX210S bound to program CBX210 in library NEWLIB, the following command will do.

```
CHGPGMDTA OBJ(NEWLIB/CBX210)
          BNDSRVPGM(CBX210S)
          CHGPARM(*BNDSPGLIB *ALL *BLANK *BLANK NEWLIB *BLANK
*BLANK))
```

Note that in order to update a service program's library qualification, the program referring the service program must have been created with the Allow *SRVPGM library update (ALWLIBUPD) attribute set to *YES. As suggested by the above command example, you can target a current library value of *LIBL by the command's *ALL library special value. While this allows for *LIBL library qualifications to be changed to a specific library name, there's no support for the reverse change of a specific library name to the special value *LIBL. Although the API documentation specifies *LIBL as a valid *To library* name, the QBNCHGPD API did not accept this value when I first tested the CHGPGMDTA command processing program. Raising the issue with IBM returned the reply that the documentation, not the API, was wrong. Consequently, IBM intends to correct the QBNCHGPD API documentation at some point in the future.

In the following example, I change the library name of the service program export source file library name for the specified service program from *LIBL to library name NEWLIB:

```
CHGPGMDTA OBJ(NEWLIB/CBX210S)
          OBJTYPE(*SRVPGM)
          CHGPARM(*EXPSRCFIL *ALL QSRVSR CBX200B NEWLIB))
```

The above examples rely on specifying and targeting a single program data reference, but using the an appropriate combination of the CHGPGMDTA command's To and From library, file, and member parameters would allow you to also change multiple program data references with a single command execution. If a program, for example, is created from a number of modules, all having been moved from library OLDLIB to NEWLIB, the following command would do the job:

```
CHGPGMDTA OBJ(NEWLIB/CBX210)
          BNDMODULE(*ALL)
          BNDSRVPGM(*ALL)
          CHGPARM(*BNDMODLIB OLDLIB *BLANK *BLANK NEWLIB *BLANK
*BLANK))
```

In order to fully comprehend what the CHGPGMDTA command is capable of--and what it's not capable of--you would probably find it helpful to conduct a few tests on your own. In order to avoid irreversible actions and unforeseen complications involving production objects, I suggest you perform the test against program and module objects copied to a safe test environment. And if you have any questions, feel free to contact me at flensburg@novasol.dk.

This APIs by Example includes the following sources:

```
CBX210    -- RPGLE    -- Change Program Data
CBX210H   -- PNLGRP   -- Change Program Data - Help
CBX210V   -- RPGLE    -- Change Program Data - VCP
CBX210X   -- CMD      -- Change Program Data

CBX210M   -- CLP      -- Change Program Data - Build command
```

To create all these objects, compile and run the CBX210M program, following the instructions in the source header. As always, the compilation instructions are also included in the respective source headers.

QBNCHGPD API PTF Cover Letters:

[QBNCHGPD API 5.4 PTF SI28759 \(superseded by SI32007\)](#)

[QBNCHGPD API 5.4 PTF SI32006:](#)

[QBNCHGPD API 5.4 PTF SI32007](#)

IBM Software Knowledge Base Documents:

[API QBNCHGPD \(Change Program Data\) will Update Bound Module Library Name](#)

[New API QBNCHGPD in v6r1](#)

This article demonstrates the following Program and CL Command API:

[Change Program Data \(QBNCHGPD\) API](#)

[Program and CL Command APIs \(Release 6.1\)](#)

You can retrieve the source code for this API example from:

http://www.pentontech.com/IBMContent/Documents/article/58690_1175_ChgPgmDta.zip.

Source URL: <http://iprodeveloper.com/rpg-programming/apis-example-release-61-change-program-data-api-ptfed-54>