

APIs by Example: Change and Retrieve Command Exit Points - and Command Exit CL Commands

[System iNetwork Programming Tips Newsletter](#)

[Carsten Flensburg](#)

Carsten Flensburg

Thu, 03/25/2010 (All day)



The Registration Facility on your IBM i provides a multitude of exit points that let you integrate system functions with your own programs and tools by adding exit programs to these exit points. You can develop the exit programs yourself, or for some exit points, third-party products are available. Some of the more common examples of the latter are the products that monitor and control access to the client connections on your IBM i, such as FTP, Telnet, ODBC, and many others. To get an overview of all the exit points available, run the command Work with Registration Information (WRKREGINF).

Today's APIs by Example focuses on two distinct exit points: Change Command and Retrieve Command Exit Points. The former enables you to register a single exit program for almost any CL command on your system. The latter provides for up to 10 exit programs being registered for each CL command. There are a few commands, mainly CL programming ones, for which exit programs can't be registered. CL commands to add and remove exit programs to specific exit points and exit point formats are included with the system, as are corresponding Registration Facility APIs. With this article, I'm also adding three CL commands of my own invention to offer a dedicated support of the command exit points mentioned.

The two command exit points, QIBM_QCA_CHG_COMMAND and QIBM_QCA_RTV_COMMAND, were added to the System i Registration Facility with release 4.5. They offer a robust and documented approach to replace risky methods developed over time to, for example:

- modify system commands using the CHGCMDDFT command
- create and rename copies of system commands
- keeping modified copies of system commands in a library ahead of QSYS
- prevent specific command parameter values by attaching a user-written command validity checker to a system command or copy

The command exit points are interesting to programmers, because in addition to developing exit programs enforcing system security and configuration for the system administrators, they also let us create exit programs that adapt the commands being used as part of our daily job. One of the exit program examples included with this article, for example, ensures that all modules and programs are created with the appropriate debugging information required to investigate and debug incidents as well as recover program source, both during the development stage and later on in the production environment.

The Retrieve Command Exit Point programs are called immediately before control is passed to the CPP, and the exit programs registered for the command are called in the sequence defined by the exit program number, lowest first, if more than one exit program was registered for the command. The

Retrieve Command Exit Point allows you to realtime monitor the execution of the registered command as well the command parameters specified.

This allows you to, for example, audit commands based on the values submitted for the command's parameters or to launch other commands and perform environment configuration triggered by specific commands and the parameters entered. I've included a couple of exit programs with this article to demonstrate situations in which this type of command exit point comes in handy.

The Change Command Exit Point program registered for a command, as the name suggests, allows you to actually change the command or its parameters and is called once before any of the four following events occur:

- the command is prompted
- interparameter checks are performed
- the command validating checking program is called
- control is transferred to the CPP

So if, for example, the command is being prompted, the exit program is called only immediately before the command prompt panel is displayed. It's up to the exit program developer to ensure that any of the changes in the command's parameters performed by the exit program are protected from being changed during the command prompt. This protection is achieved using selective prompt characters that must be added to the parameter keyword(s) in question. I've included a link to selective prompt character documentation at the end of this article, in case you need to brush up on this topic.

The Change Command Exit Point is always processed before the Retrieve Command Exit Program, and a quite complex algorithm is applied in order to control the execution path and extent of the events taking place. I'll try to sum up the most significant rules here. The Change Command Exit Point program can change any of the following parts of the command string being passed to it:

- the command name
- the command library
- the command's parameters

There are, however, exceptions to this capability; changes to the command will be prohibited if the command being executed

- is specified in qualified format with a specific library name, as in the following example:
QSYS/STRPRTWTR
- has a parameter defined as RTNVAL(*YES), allowing the CPP to return a value to its caller.
- has a parameter defined as DSPINPUT(*NO) or DSPINPUT(*PROMPT), preventing the parameter value from being shown in the prompt panel.
- was run in a program running in system state.

IBM a while back introduced a command qualification concept implemented by the special values *SYSTEM and *NLVLIBL, which are employed in system programs, menus, and functions in order to locate the command qualified by these special values. The *SYSTEM special value points to system library QSYS exclusively, and the *NLVLIBL special value points to the national language version (NLV) libraries in the job library list plus system library QSYS. The *SYSTEM and *NLVLIBL special values can be used in nonsystem CL programs also. Commands qualified by any of these two special values are still subject to change by the change command exit point.

A side note: Registering the Replace Command Exit Program (QCARPLCM) API for a command as a change command exit program will cause the full job library list to be used to locate a command qualified by either *SYSTEM or *NLVLIBL, in effect reverting to the previous behavior of a nonqualified command, letting you override system commands to a copy in a library ahead of QSYS in the system library list.

If the command string is changed by the change command exit program the command analyzer will discontinue the current execution path and start the process over again with the replaced command string, including a full validation of the new command string and prompting the command string if requested.

If a different command in a different library replaced the original command, the Change Command Exit Point will be called for the new command, if registered, but the exit point will not allow the exit program to change the command string. If the new command is registered with the Retrieve Command Exit Point, the exit programs applying for this exit point will also be called.

For commands entered on a command line, the original command will be logged as a request message, while the new command will be logged as a command message. Both strings will be eligible for retrieval with the function key F9=Retrieve command function. Note that any escape messages sent by either exit points' exit programs will be ignored by the command analyzer. They will be left in the job log, and normal command processing will continue.

As documented above, the Change Command Exit Point enables you to intercept commands being executed and, provided no restrictions are met, to alter both the command and its parameters, as in the following examples:

- Change the command's parameters to ensure specific values being included
- Restrict certain parameter values by changing them to original values
- Change the command name and its parameters entirely
- Change the command library

If restrictions preventing you from changing the command string apply, you still have the option of monitoring and reporting such events to follow up on and, if appropriate, correct.

Once you've decided to register an exit program for a specific command, you'll want to prompt the Add Exit Program (ADDEXITPGM) command as in the example below, which will register the exit program CBX213X1 in library QGPL for the CRTCLPGM in library QSYS for the Change Command Exit Point:

```
AddExitPgm  ExitPnt( QIBM_QCA_CHG_COMMAND )
              Format( CHGC0100 )
              PgmNbr( *LOW )
              Pgm( QGPL/CBX213X1 )
              PgmDta( *JOB 20 'CRTCLPGM  QSYS      ' )
```

While the above ADDEXITPGM command by no means constitutes a roadblock in terms of skills required, I still decided to take advantage of the Registration Facility APIs to create my own version, specifically targeting the change and retrieve command exit points. This also gave me an opportunity to provide an example of the Add Exit Program (QusAddExitProgram) API. Here's the alternative and slightly simpler version of the previous command exit registration:

```
AddCmdExit CmdExit( *CHG )
              Cmd( QSYS/CRTCLPGM )
              ExitPgm( QGPL/CBX213X1 )
```

The ADDCMDEXIT command also provides access to the additional ADDEXITPGM command parameters, as documented by the command's prompt panel displayed here:

Add Command Exit Program (ADDCMDEXIT)			
Type choices, press Enter.			
Command exit type		*CHG, *RTV	
Command		Name	
Library		Name, *CURLIB	
Exit program		Name	
Library		Name, *CURLIB	
Program number	*LOW	1-2147483647,	
	*HIGH		
Text 'description'	*BLANK		
Threadsafe	*UNKNOWN	*UNKNOWN, *NO,	
		*YES	
Multithreaded job action	*SYSVAL	*SYSVAL, *RUN,	
		*MSG, *NORUN	
Additional Parameters			
Replace existing entry	*NO	*YES, *NO	

The command and its parameters are fully documented by a command help text panel group. As for registering exit programs against the command exit point, I should point out that the introduction of proxy commands in release 5.4 had a severe impact on the command exit point program registration rules. A proxy command in essence provides a shortcut to a command; it contains no other definitions but the name of the target command.

IBM compiler products such as the ILE RPG, RPG/400, OPM & ILE COBOL, and ILE C compilers are all packaged in product libraries, and consequently all CRTRPG*, CRTCBL*, CRTCL*, as well as CRTBND* and CRTSQL* commands are located in their appropriate product libraries. Prior to release 5.4 the product installation process would create copies of all these product commands in library QSYS. As of release 5.4, however, the QSYS compiler command copies all were converted to proxy command versions.

Since the command exit points don't support proxy commands, the release 5.4 change caused all previously registered QSYS compiler commands to cease having the associated exit program called. To reinstate the pre-5.4 behavior, you'd need to register the "regular" command at the end of the proxy chain instead of the QSYS proxy. All this was fully documented in the Memo to Users V5R4 so in itself the challenge was somehow surmountable, had it not been for the fact that many systems also have secondary languages (NLV) installed. The NLV QSYS29nn libraries still hold regular copies of the compiler commands, but the command exit points prohibit the registration of commands residing in an NLV library.

IBM therefore decided to release PTFs for releases 5.4 and 6.1 to maintain the pre-5.4 behavior of a single exit point being valid for both primary and secondary languages. Following the installation of the appropriate PTF, you'll be allowed to register proxy command residing in the QSYS library, and hence overcome the restriction and inconvenience caused by the original implementation of proxy commands. I've included links at the end of this article to the APARs describing this issue in detail and providing links to the PTFs, in case they are missing on your system.

Anyway, to remove a previously registered command exit program, you use the Remove Exit Program (RMVEXITPGM) command, as demonstrated here:

```
RmvExitPgm ExitPnt( QIBM_QCA_CHG_COMMAND )
              Format( CHGC0100 )
              PgmNbr( 1 )
```

This command would remove the exit program registered with sequence number 1 for the Change Command Exit Point. Again, I decided to offer a simplistic version based on the Remove Exit Program (QusRemoveExitProgram) API. Here's the Remove Command Exit Program (RMVCMDEXIT) command's prompt panel:

Remove Command Exit Program (RMVCMDEXIT)

Type choices, press Enter.

Command exit type	*CHG, *RTV
Program number	1-2147483647,
*LOW, *HIGH	

Please refer to the command help text panel group for all the details. Using the RMVCMDEXIT command, the task of removing the Change Command Exit Point's exit program having sequence number 1 would require the following command input:

```
RmvCmdExit CmdExit( *CHG )
              PgmNbr( 1 )
```

If you over time add more command exit programs than you're removing, you'd at some point welcome a command providing you with an overview of, and a management interface for, all your change and retrieve command exit programs. Again, the i OS provides for a command that will support this requirement. To get a list of all the exit programs registered for the change command exit program you'd run the following command:

```
WrkRegInf ExitPnt( QIBM_QCA_CHG_COMMAND )
```

From the *Work with Registration Information* list panel, you'd then select option 8=*Work with exit programs* and subsequently be presented with a list displaying all exit programs registered. Entering option 5=*Display* for a list entry would take you to a panel revealing for which command the exit program is in effect. Alternatively, you could employ the Retrieve Exit Information (QusRetrieveExitInformation) API to get at the exit point and exit program information needed in order to provide a work-with interface specifically adapted to the command exit points. Here's how the Work with Command Exit Programs (WRKCMDEXIT) command prompt looks:

Work with Command Exit Pgms (WRKCMDEXIT)

Type choices, press Enter.

Command exit type	*CHG	*CHG, *RTV
Select:		
Command	*ANY	Name, *ANY
Library	*ANY	Name, *ANY
Exit program	*ANY	Name, *ANY
Library	*ANY	Name, *ANY

You specify which of the two exit points you want to work with and optionally a set of selection criteria allowing you to limit the list to certain commands, exit programs or libraries. Help text is available to provide further command and parameter documentation. Below you see an example of what the initial list view looks like on my system:

Work with Command Exit Programs

WYNDHAMW

17-03-10

08:52:22

Command exit . . . : QIBM_QCA_CHG_COMMAND

Text : Change command exit programs

Type options, press Enter.

2=Replace

4=Remove

5=Work with program

8=Work with command

Exit

Opt	Command	Library	Text
Format			
	CRTRPGPGM	QSYS	Create RPG/400 Program
CHGC0100			
	CRTCLPGM	QSYS	Create CL Program
CHGC0100			
	CRTRPGMOD	QSYS	Create RPG Module
CHGC0100			
	CRTBNDRPG	QSYS	Create Bound RPG Program
CHGC0100			
	CRTCLMOD	QSYS	Create CL Module
CHGC0100			
	CRTCMOD	QSYS	Create C Module
CHGC0100			
	CRTCBLMOD	QSYS	Create COBOL Module
CHGC0100			

Bottom

Parameters or command

===>

F3=Exit

F4=Prompt

F5=Refresh

F9=Retrieve

F11=View 2

F12=Cancel

F21=Print list

F22=Display entire field

F24=More keys

The list panel has a second view to and from which you can toggle using command key F11. The alternate view displays the exit program name and library, the exit program number, and the text description of the exit program. Cursor-sensitive help text is included for the entire list panel as well

as its separate parts. Note that due to restrictions by default applying to the underlying APIs, the ADDCMDEXIT and the RMVCMDEXIT commands both require *ALLOBJ and *SECADM special authority to run successfully.

This article has focused on the command exit points, but in case you'd like to investigate the many other exit points available, I've added links below to a wealth of articles discussing a broad range of the exit points available on IBM i and providing detailed examples of exit programs for these exit points.

This APIs by Example includes the following sources:

```
CBX213  -- RPGLE  -- Work with Command Exit Programs

CBX213E -- RPGLE  -- Work with Command Exit Programs - UIM General
Exit Program
CBX213H -- PNLGRP -- Work with Command Exit Programs - Help

CBX213P -- PNLGRP -- Work with Command Exit Programs - Panel Group

CBX213X -- CMD    -- Work with Command Exit Programs


CBX2132 -- RPGLE  -- Add Command Exit Program - CPP
CBX2132H -- PNLGRP -- Add Command Exit Program - Help

CBX2132V -- RPGLE  -- Add Command Exit Program - VCP
CBX2132X -- CMD    -- Add Command Exit Program


CBX2133 -- RPGLE  -- Remove Command Exit Program - CPP
CBX2133H -- PNLGRP -- Remove Command Exit Program - Help
CBX2133X -- CMD    -- Remove Command Exit Program


CBX213M -- CLP    -- Command Exit Programs - Build commands
```

To create all Command Exit Program command objects, compile and run the CBX213M program, following the instructions in the source header. You can also find compilation instructions in the respective source headers. As mentioned, I've also included the following examples of exit programs that can be registered for the change and retrieve command exit points:

```
CBX213X1 -- RPGLE  -- Change compile commands to include debug info
CBX213X2 -- RPGLE  -- Change signoff command to list job log
CBX213X3 -- CLP    -- Query/400 printer file override exit program
CBX213X4 -- RPGLE  -- Change device description prior to STRPRTWTR
```

Please refer to the source header for information on how to create and register the above programs to the appropriate command exit point.

IBM Documentation:

[Command Analyzer Change Exit Program](#)

[Command Analyzer Retrieve Exit Program](#)

[Exit Programs Concept](#)

[Replace Command Exit Program \(QCARPLCM\) API](#)

[Using Selective Prompting for CL Commands](#)

[Example of Program for Exit Point QIBM_QCA_RTV_COMMAND](#)

[Rebuild Registration Facility Repository](#)

[APAR SE32544 \(5.4\): ERROR REGISTERING QSYS PROXY COMMANDS](#)

[APAR SE32544 \(6.1\): ERROR REGISTERING QSYS PROXY COMMANDS](#)

Exit Point Articles:

[A Change in Fundamentals: Exit Points for i/OS AS/400 Control Language Commands](#)

[Exit Points for AS/400 Control Language Commands: Auditing Command Usage in Real Time](#)

[Enhance Security with CL Command Exit Point Programs](#)

[What's All This About Exit Points and Exit Programs?](#)

[The Wonders of the Telnet Exit Program](#)

[APIs by Example: Profile Exit Programs \(QWTSETPX/QWTRTPPX\)](#)

[APIs by Example: Realtime Job Monitor](#)

[Preventing Immediate Deletion of Journal Receivers](#)

[Notification When Critical ASP Usage Limits Occur](#)

[Synchronizing Passwords Between Systems](#)

[Sending Commands to Another Job - Revisited for i5/OS V5R4](#)

[Additional Validation of New Passwords](#)

[Password Change Blocking for V5R4 and Earlier](#)

[Query Governor Control System for V5R4](#)

[Printing Your Registered Exit Programs Nicely!](#)

This article demonstrates the following Registration Facility APIs:

[Add Exit Program \(QusAddExitProgram\) API](#)

[Remove Exit Program \(QusRemoveExitProgram\) API](#)

[Retrieve Exit Information \(QusRetrieveExitInformation\) API](#)

[Registration Facility APIs](#)

[Retrieve the source code for this API example.](#)

Source URL: <http://iprodeveloper.com/rpg-programming/apis-example-change-and-retrieve-command-exit-points-and-command-exit-cl-commands>