


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

APIs by Example: Adding to the Collection of Validation List Commands

[System iNetwork Programming Tips Newsletter](#)

[Carsten Flensburg](#)

Carsten Flensburg

Thu, 07/23/2009 (All day)

Last time, I presented four new CL commands covering a validation list entry's life cycle in terms of validation list entry management. [If you missed that article, now is the time to go back and learn more about the ADDVLDLE, VFYVLDLE, CHGVLDLE, and RMVVLDLE commands.](#) As I mentioned in the closing paragraph of that article, today's focus will be on the validation list object itself, as well as on a couple of new CL commands that ease the use and management of validation lists.

If you're on an IBM i command line, run the command GO CMDVLDL, and you'll see that the operating system includes two native validation list CL commands: Create Validation List (CRTVLDL) and Delete Validation List (DLTVLDL). Programming for and in the context of working with validation lists, this offer is at least two commands short. So to fill that gap, today's APIs by Example adds two new commands to the collection: Work with Validation List (WRKVLDL) and Clear Validation List (CLRVLDL). Today's article also gives me an opportunity to briefly discuss the validation list object and how this object type is implemented on the IBM i.

As for the latter, validation lists are in the perspective of the system an object of internal type x'oE', or in other terms an *independent index*, a property it shares with many other IBM i objects. It's the object's internal subtype x'10' that identifies an object as a validation list. Other prominent independent index-based external object types include:

Object	Subtype	Symbolic Object type
Job queue	x'01'	*JOBQ
Output queue	x'02'	*OUTQ
Message file	x'03'	*MSGF
User index	x'10'	*USRIDX
Job scheduler	x'0C'	*JOBSCD

Here are examples of internal independent indexes employed by and directly accessible only to the operating system (except for independent index MI instructions at security level 30 and below):

Object	Subtype	Context	Index content
'User profile'	x'C4'	QSYS	User profile command defaults
QSYUPTBL	x'C5'	QSYS	User profile table
'User ID' 'Address'	x'D1'	QUSRSYS	User ID/Address network files

Provided that you have the necessary user profile special authority of *ALLOBJ, you can use the Dump System Object (DMPSYSOBJ) command to list the attributes and content of independent indexes to a spooled file, and then use this as a starting point for your investigation of the content and functionality of independent indexes. Here's how you'd dump the network files for user ID JIMC at address SYSTEMA:

```
DMPSYSOBJ OBJ('JIMC SYSTEMA') CONTEXT(QUSRSYS) TYPE(0E) SUBTYPE(D1)
```

Note the four blanks between JIMC and SYSTEMA; each element of the user ID and address value must occupy 8 bytes to fit the index entry's key format. The following MI instructions are available to access and manipulate independent indexes directly:

MI Instruction	Name
CRTINX	Create Independent Index
DESINX	Destroy Independent Index
FNDINXEN	Find Independent Index Entry
INSINXEN	Insert Independent Index Entry
RMVINXEN	Remove Independent Index Entry
MATINXAT	Materialize Independent Index Attributes
MODINX	Modify Independent Index

C library functions also exist for all the above MI instructions. They are located in service program QC2UTIL1 and are easily accessible through binding directory QC2LE.

I mention above that the system security level is involved in the equation as far as actually using the above MI instructions or C library functions is concerned. The condition to observe is defined by the independent index object's domain attribute. At security level 40 and 50, accessing an object in the *SYSTEM domain is only possible to a *SYSTEM state program. If a *USER state program tries to access a *SYSTEM domain object at security level 40 or above, exception MCH6801 is issued: **Object domain or storage protection error for offset &5 in object &1.**

At release level 5.4 and earlier, access to the system service tools through the Start Service Tools (STRSST) command would allow you to patch a *USER state program to become *SYSTEM state. Unless you also recalculated and updated the program validation value (program checksum), the system would know that the program object had been tampered with, but the program would run. Given a security audit level (QAUDLVL) of *AUTFAIL, security audit entries of type AF will, however, be deposited for each call to a patched program. As of release 6.1, IBM has blocked the possibility of running a patched program, so at security level 40 or 50 there's no longer a way around this restriction.

Given the [recommendation](#) of setting the system security level to a minimum of 40, you therefore have to rely on public interfaces in the form of system APIs to access *SYSTEM domain object information when you upgrade to release 6.1. More about accessing independent index type objects in a minute.

Now it's time to present the first of the two new validation list CL commands. The WRKVLDL command offers the following simple interface:

Work with Validation Lists (WRKVLDL)

Type choices, press Enter.

Validation list	Name, generic*, *ALL
Library *LIBL	Name, *LIBL, *CURLIB...
Sort order *VLDL	*VLDL, *LIB

You enter a combination of a name, generic name, or the special value *ALL for the validation list and a library name or any of the library list special values to select the validation lists to include in the work with list panel. You can also specify the order in which the list is presented, either validation list name or library name. Note

that the WRKVLDL command can be run only in an interactive job. Please refer to the online command help text panel group for all the details.

Once you've installed the WRKVLDL command, to see the list of validation lists and the functions available from the Work with Validation Lists display panel, run the following command:

```
WRKVLDL  VLDL (QUSRSYS/*ALL)
```

With a bit of luck, the following list panel or similar, depending on the actual release and configuration of your system, should be displayed:

Work with Validation Lists				
WYNDHAMW			18-07-09	
18:18:44				
List order . . . :		*VLDL	Position to . . .	
Type options, press Enter.				
2=Edit authority		3=Copy	4=Delete	7=Rename 8=Work with object
13=Change description		14=Clear		
Validation		Entry		
Opt	List	Library	Count	Text
	QGLDVLDL	QUSRSYS	4	
	QSASRVID2B	QUSRSYS	7	
	QSJSRVID	QUSRSYS	1	
	QTOCPTP	QUSRSYS	5	
	QTOCSASVL	QUSRSYS	7	
	QTOVDVPKEY	QUSRSYS	2	
	QTOVDVSKEY	QUSRSYS	0	
	QTOZRADI	QUSRSYS	0	
Bottom				
Parameters or command				
===>				
F3=Exit		F4=Prompt	F5=Refresh	F6=Create validation list
F9=Retrieve				
F11=Display full text		F12=Cancel	F17=Top	F18=Bottom

A number of options are available to perform against the selected validation lists, such as editing authority or copying, deleting, renaming, or clearing the validation list(s) of your choice. You can also position the list to a specified item, being either a validation list name or library name, depending on the list order specified on the WRKVLDL command. A single option is, however, currently missing, namely option 12, which will let you work with a validation list's entries. In the next APIs by Example article, I'll present the Work with Validation List Entries (WRKVLDLE) command, and by that time, once the WRKVLDLE command is installed, option 12 will surface on the above list panel.

For performance reasons, I use the `matinxat()` - Materialize Index Attributes function mentioned earlier to determine the number of validation list entries if the WRKVLDL command is run on a system at security level 30 or lower. Since the `matinxat()` function depends on a system pointer to the independent index object in question in order to return the index attributes, the `rslvsp()` - Resolve System Pointer function is used to fulfill this requirement.

Note that RPG IV has no explicit support of system pointers, but defining an uninitialized procedure pointer will make the RPG compiler create an open pointer, capable of storing any type of pointer—including a system pointer. This capacity is exploited by the CBX2061 program to allow for the "cheap" access to the validation list attribute information at security level 30 or below. If you don't appreciate using undocumented features, you can change the program to the other approach supported, as explained below.

For systems at a higher security level, I use the Open List of Validation List Entries (QSYOLVLE) API to retrieve the total number of index entries, but since this approach involves building a list of all the current validation list entries, it's potentially significantly slower, depending on the number of validation list entries currently held in the validation list. Feel free to eliminate the entry count information to save the processor cycles if you're at security level 40 or 50 and find this information of little value.

Of the options currently present on the Work with Validations Lists panel, only option 14=Clear is not evoking a native IBM i CL command. The CLRVLDL command is also included with today's article and has the following command prompt:

Clear Validation List (CLRVLDL)

Type choices, press Enter.

Validation list	Name
Library	*LIBL Name, *LIBL, *CURLIB

You specify the qualified name of the validation list that you want to clear and press Enter. Due to the possible dire consequences of clearing a validation list, special authority *ALLOBJ and *SECADM is required for the user profile running the CLRVLDL command. If the command is run in an interactive job, there's also a final warning message window asking you to confirm the intention of clearing the specified validation list. Running the command:

```
CLRVLDL VLDL (QPGL/WEBPWD)
```

from a command line will cause the following message window to appear:

```
:          Clear Validation List (CLRVLDL) :  
: You have requested that validation list WEBPWD in library QGPL :  
: should be cleared. Be advised that this is an irreversible action, :  
: unless a current and fully functional back-up of the validation list :  
: is available. To confirm this request press Enter. To cancel this :  
: request press F12. :  
: :  
: : Bottom :  
: F12=Cancel :  
: :  
: ..... :
```

As the warning explains, pressing F12 will cause the command processing to terminate immediately. Pressing Enter, however, will cause the specified validation list to be cleared without further notice. This function is based on the ability of the Change Current Job (QWCCCJOB) API to intercept an Exit (F3) or Cancel (F12) function key being pressed in a job together with the option of using the Retrieve Current Attributes (QWCRTVCA) API to reset these job attributes. Please check out program CBX2061 for the implementation details.

This APIs by Example includes the following sources:

```

CBX2061  -- RPGLE  -- Work with Validation Lists - CCP
CBX2061E -- RPGLE  -- Work with Validation Lists - UIM Exit Program
CBX2061H -- PNLGRP -- Work with Validation Lists - Help
CBX2061P -- PNLGRP -- Work with Validation Lists - Panel Group
CBX2061V -- RPGLE  -- Work with Validation Lists - VCP
CBX2061X -- CMD     -- Work with Validation Lists

CBX2062  -- RPGLE  -- Clear Validation List - CPP
CBX2062H -- PNLGRP -- Clear Validation List - CPP
CBX2062V -- RPGLE  -- Clear Validation List - VCP
CBX2062X -- CMD     -- Clear Validation List

CBX206M  -- CLP     -- Validation List Commands - Build commands

```

To create all these validation list command objects, compile and run CBX206M, following the instructions in the source header. Note that for the CL program CBX206M to run successfully and for the new validation list command objects to compile, it is required that the CBX205M CL program included with [the previous APIs by Example article of June 25](#) has also been compiled and run successfully. As always, the compilation instructions are also included in the respective source headers.

If you run at security level 30 (but hopefully not lower) and are interested in learning more about the external object types based on independent indexes as well as the MI functions available to access this object type, you might like to know that I've written a Display Index Entry Count (DSPIDXECNT) command that documents all the symbolic object types and details the programming steps of accessing the index and its attributes. Due to the security level restriction, I excluded the DSPIDXECNT command from this article, but if you want a copy, please send me an [email](#).

IBM Documentation:

Validation List Objects

Planning the Use of Validation List Objects

Validation list on HTTP Server

Previously published related articles:

[APIs by Example: A Validation List Entry's Life Cycle in CL Commands](#)

[APIs by Example: Have a Peek at Validation List Entries](#)

[APIs by Example: User Function Registration APIs, Part 1](#)

[APIs by Example: User Function Registration APIs, Part 2](#)

[APIs by Example: User Function Registration APIs, Part 3](#)

[APIs by Example: Validation List APIs](#)

[APIs By Example: Profile Authorization Management](#)

[APIs by Example: Cryptographic Services APIs, Part 3](#)

[APIs by Example: Cryptographic Services APIs, Part 7](#)

C library functions demonstrated (MI documentation):

[rslvsp\(\)- Resolve system pointer - MI built-in RSLVSP](#)

[matinxat\(\) - Materialize independent index attributes - MI built-in MATINXAT](#)

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

Source URL: <http://iprodeveloper.com/rpg-programming/apis-example-adding-collection-validation-list-commands>