

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

APIs by Example: Have a Peek at Validation List Entries

[System iNetwork Programming Tips Newsletter](#)

[Carsten Flensburg](#)

Carsten Flensburg

Thu, 05/28/2009 (All day)

I've discussed and demonstrated validation lists in a number of articles and utilities previously published in this newsletter. On these occasions, I've used validation lists to store sensitive session information as well as cryptographic encryption keys. Basically though, you can think of validation lists as password files, storing a user ID and an encrypted password, and as such I've also successfully exploited validation lists when developing web applications requiring a secure authorization mechanism. In addition to user ID and password, a few other validation list attributes are supported--more about this in a minute.

Although IBM provides native CL commands to create and delete a validation list object, you have to resort to APIs when it comes to creating, changing, retrieving, verifying, or deleting validation list entries. On the one hand, this constraint makes it a bit easier to control who is accessing validation list entries because programming skills are involved, but on the other hand things are also a bit more complicated when it comes to testing and debugging your applications that take advantage of validation lists--not to mention the efforts involved in the administration of the validation list entries. So why not create a few CL commands to close the gap, the first of which is presented in today's APIs by Example.

Let's start with a brief description of the validation list entry. A validation list entry is divided into three directly accessible main attributes:

1. Entry ID (up to 100 bytes)
2. Entry data (up to 1000 bytes)
3. Encrypted data (up to 600 bytes)

Each of the above attributes also have a related Coded Character Set Identifier (CCSID) attribute available to identify the CCSID, if any, of the value stored in the *Entry ID*, the *Entry data* and the *Encrypted data*, respectively. Also, the length of each of the attributes must be stored with it, and this requirement has the impact that trailing blanks in the Entry ID, if included in the length specification, actually must be included again when you try to locate this validation list entry. As the example in the Validation List API manual demonstrates this property, "Smith" and "Smith " are considered different Entry IDs due to the difference in length. As alleged, the *Entry ID* is where you'd store a user name or identifier, and it constitutes the entry key when you need to retrieve, verify, or delete the validation list entry.

The *Entry data* attribute simply offers a space to store unformatted data, a data structure, or for example a text string describing the entry. The *Encrypted data* can be stored in one of two ways, depending on the value of the system value QRETSVRSEC (retain server security data). If the QRETSVRSEC system value is 0 (zero), the encrypted data is only one-way encrypted, so that the clear value can't be retrieved again. You can still verify the encrypted value, though, because similar to the System i password verification method, the value to compare against is simply encrypted using

the same encryption key prior to the verification process, so that the *encrypted values* are compared during this process.

If the QRETSVRSEC system value is set to 1 (one), however, the choice is yours, and the encrypted data retrieval option is defined by the QsyEncryptData attribute value submitted to the API, when you add or change the validation list entry. If all you need is to be able to verify a password, the one-way encryption is of course the safest method to apply. If you need to be able to access the encrypted data again, as I did in the two examples referred to above, setting the QsyEncryptData attribute to '1' will enable you to do so.

If you've stored the encrypted data in a decryptable form, a CL command is actually available (as of release 5.2) to allow you to remove it, system wide though. The Clear Server Security Data (CLRSVRSEC) command lets you clear all decryptable authentication information associated with user profiles and validation list (*VLDL) entries on your system. Prior to release 5.2, all this information was implicitly removed when the QRETSVRSEC system value was changed from 1 to 0. Performing such a change on subsequent releases only prohibits the retrieval of the decryptable data; the data remains in the validation list entry until the CLRSVRSEC command is run; it just can't be retrieved by any means as long as QRETSVRSEC is 0. Once QRETSVRSEC is changed back to 1, the encrypted data will again be retrievable, provided that the CLRSVRSEC command was not run in the mean time.

Validation list entries also contain information maintained by the system and enabling you to establish and enforce password security and policy. The following four events are recorded and immediately available by using the appropriate API:

1. Validation list entry create timestamp
2. Encrypted data change timestamp
3. Encrypted data last verified timestamp
4. The count of invalid verification attempts since last successful

It is, however, up to you to actually take advantage of this information and put the necessary control and monitoring facilities in place.

As far as creating, accessing, and deleting validation list entries, quite a few APIs are available in both OPM and ILE versions. Here's the list for release 5.4, and I've removed all duplicate OPM APIs:

- Add Validation List Entry (QsyAddValidationLstEntry())
- Change Validation List Entry (QsyChangeValidationLstEntry())
- Convert Validation List (QSYCVTVL)
- Find First Validation List Entry (QsyFindFirstValidationLstEntry())
- Find Next Validation List Entry (QsyFindNextValidationLstEntry())
- Find Validation List Entry (QsyFindValidationLstEntry())
- Find Validation List Entry Attributes (QsyFindValidationLstEntryAttrs())
- Open List of Validation List Entries (QSYOLVLE)
- Remove Validation List Entry (QsyRemoveValidationLstEntry())
- Verify Validation List Entry (QsyVerifyValidationLstEntry())
- Add Validation List Certificate (QsyAddVldlCertificate)
- Check Validation List Certificate (QsyCheckVldlCertificate)
- List Validation List Certificates (QsyListVldlCertificates)
- Remove Validation List Certificate (QsyRemoveVldlCertificate)

For the purpose of displaying a known validation list entry and its attributes, the Find Validation List Entry (QsyFindValidationLstEntry()) and Find Validation List Entry Attributes (QsyFindValidationLstEntryAttrs()) APIs do the job nicely. To show you how, I've created the Display Validation List Entry (DSPVLDLE) command and the associated command processing program. Here's how the command prompt looks:

```

                                Display Validation List Entry (DSPVLDLE)

Type choices, press Enter.

Validation list . . . . . Name
Library . . . . . *LIBL Name, *LIBL,
*CURLIB
Entry ID . . . . .
Coded character set identifier *DFT 1-65534, *DFT,
*HEX
Output format . . . . . *CHAR *CHAR, *HEX
Output . . . . . * *, *PRINT

```

The qualified name of the validation list and the entry ID are the command's key parameters, uniquely identifying the validation list entry to display. The *Coded character set identifier* defines the CCSID of the entry ID and is as such not directly used as a search argument but merely identifies the CCSID of the provided entry ID. You can choose to display the entry ID, entry data, and encryption data (if available) in either character or hexadecimal format, depending on the special value provided for the *Output format* (OUTFMT) parameter. And finally, the *Output* (OUTPUT) parameter decides whether the validation list entry should be displayed or printed. An online help text panel group is provided to explain all command parameters in detail.

Below is an example of how entry ID "o " (8 trailing blanks) in the QSASRVID2B validation list looks on my system. Here's page 1:

```

                                Display Validation List Entry

WYNDHAMW
23-05-09
19:41:42
Validation list . . : QSASRVID2B

Library . . . . . : QUSRSYS

```

```
Entry ID . . . . . :    0

CCSID . . . . . :    37

Length . . . . . :    9

Entry data . . . . . :    940665-58CAB V5R2M0P20003IQ2004-
03-13-16.06.02
810

CCSID . . . . . :    37

Length . . . . . :    50

More...
Press enter to continue.

F3=Exit   F5=Refresh   F12=Cancel   F22=Display entire field value
```

Page down and see the remaining validation list entry information on page 2:

```
                                Display Validation List Entry

WYNDHAMW                                23-05-09

19:41:42
Validation list . . :    QSASRVID2B

Library . . . . . :    QUSRSYS

Encrypted data . . . . . :    oKpy#4S54$×hME[-
```

CCSID	:	37
Length	:	16
Create timestamp	:	2005-08-24-16.05.43.858
Encrypted data timestamp	:	2005-08-24-16.05.43.858
Data verified timestamp	:	*NONE
Invalid verification count	:	0
Bottom		
Press enter to continue.		
F3=Exit F5=Refresh F12=Cancel F22=Display entire field value		

Online help text is also provided for the above display panel. Simply position the cursor and press F1 to see the help text for the chosen field or panel segment. Note that the encrypted data is included in the display only if the requesting user profile has *ALLOBJ and *SECADM special authority. In addition to regular object authority to the validation list in order to access the list entries, *USE, *ADD, and *UPD data authority is also required to retrieve the encrypted data. If you want even more granular control you could employ the Function Usage facility to decide which user profiles should be allowed to display the encrypted data using the DSPVLDLE command. I've covered the Function Usage concept and APIs in earlier articles, so please look up the links below for code examples and more information on this topic.

On a related note, release 5.4 introduced a new validation list object model, which now allows a validation list to grow to a maximum size of 1TB, instead of the original 4GB validation list capacity. The IBM announcement of this change also notes that the existing entries are stored more efficiently in a 1TB validation list. To allow existing validation lists to take advantage of this improvement, IBM included the Convert Validation List (QSYCVTVL) API with the mentioned release.

While the simple API parameter structure of this API makes it straightforward to call the API from a command line, to make it even easier I've included a CL command interface (CVTVLDL) to the API as well as online help text to document the change and its implications. If object backward

compatibility is an issue for you, please note that if a validation list is converted to a 1TB validation list, it can't be saved to a release prior to 5.2. The DSPVLDLE and CVTVLDDL commands are the first of a number of validation list commands that I will be presenting in the API by Example column, look for more validation list coverage and commands in future issues of this newsletter.

This APIs by Example includes the following sources:

```
CBX204    -- RPGLE    -- Display Validation List Entry - CPP

CBX204E   -- RPGLE    -- Display Validation List Entry - UIM General Exit
CBX204H   -- PNLGRP   -- Display Validation List Entry - Help
CBX204P   -- PNLGRP   -- Display Validation List Entry - Panel Group

CBX204V   -- RPGLE    -- Display Validation List Entry - VCP

CBX204X   -- CMD      -- Display Validation List Entry

CBX204M   -- CLP      -- Display Validation List Entry - Build command

CBX2041H  -- PNLGRP   -- Convert Validation List - Help
CBX2041X  -- CMD      -- Convert Validation List
```

To create all these DSPVLDLE command objects, compile and run CBX202M, following the instructions in the source header. As always, you'll also find compilation instructions in the respective source headers, and these guidelines should help you compile the CVTVLDDL command objects as well.

IBM Documentation:

[Validation List Objects](#)

[Planning the Use of Validation List Objects](#)

[Validation list on HTTP Server](#)

Previously published related articles:

[Securing Web Servers by Environment](#)

[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](#)

This article demonstrates the following Validation List APIs:

[Convert Validation List \(QSYCVTVL\) API](#)

[Find Validation List Entry \(QsyFindValidationLstEntry\) API](#)

[Find Validation List Entry Attributes \(QsyFindValidationLstEntryAttrs\) API](#)

[Validation List APIs](#)

[Digital Certificate Management APIs](#)

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

Source URL: <http://iprodeveloper.com/rpg-programming/apis-example-have-peek-validation-list-entries>