

APIs by Example: Cryptographic Key Management – Creating, Displaying, and Deleting Key Records

[System iNetwork Programming Tips Newsletter](#)

[Carsten Flensburg](#)

Carsten Flensburg

Thu, 02/28/2008 (All day)

This column has so far demonstrated a number of cryptographic key management utilities based on equivalent Cryptographic Services Key Management APIs. Armed with these tools, you can now establish and manage a cryptographic master key table, as well as create and maintain key store files whose content is protected by the master keys. Links to the articles delivering and discussing the LODMSTK, SETMSTK, TSTMSTK, CLRMSTK, CRTKS, and TRNKS CL commands are at the end of this article, in case you missed any of them.

Today's installment of APIs by Example adds three new CL commands to this cryptographic services toolset, all providing functionality relating to the cryptographic key records that are the real purpose of creating master keys and key stores. Say hello to the Generate Key Record (GENKR), Display Key Record Attributes (DSPKRA), and Delete Key Record (DLTKR) commands. To ease access to all these cryptographic services commands, I've also included a Cryptographic Services Commands Menu (CMDCRPSRV).

A cryptographic key record contains all the information about a cryptographic key that is required for the system to use the key in a cryptographic operation, such as the encryption or decryption of data. This information includes the key type, the key size as well as the types of cryptographic operations that the key is not allowed to perform, if any.

And most importantly, the key record stores the actual cryptographic key, encrypted under the master key assigned to the key store. I explained the concept of key stores and master key protection in more detail last time -- please follow the link below to that article for more information on this topic. To ensure a correct recovery of the encrypted cryptographic key, the key record also includes the Key Verification Value (KVV) identifying the version of the master key that was used for that encryption.

Two Cryptographic Services APIs can create a key record in a key store. One is the Write Key Record (Qc3WriteKeyRecord) API, which takes an existing key value and stores it in a key store. The other is the Generate Key Record (Qc3GenKeyRecord) API, which generates a cryptographic key of a specified key type and key length, and writes it to a key store in one single operation. That's the API that I use as the foundation for the Generate Key Record (GENKR) command.

Take a look at the GENKR command prompt, which basically exposes the input parameters of the Qc3GenKeyRecord API:

```
Generate Key Record (GENKR)
```

```

Type choices, press Enter.

Key store . . . . . Name
Library . . . . . Name
Record label . . . . .
Key type . . . . . *MD5, *SHA_1,
*SHA_256...
Key size . . . . . 1-4096
Public key exponent . . . . . *NONE *NONE, 3, 65537
Disallowed function . . . . . *NONE *NONE, *ENCRYPT,
*DECRYPT...
+ for more values
Cryptographic service provider *SFWCSP *ANYCSP, *SFWCSP,
*HWCSP
Cryptographic device name . . . *NONE Name, *NONE

```

You specify the qualified name of an existing key store and a unique (within the key store) key record label of up to 32 characters in length as the primary input parameters. The command offers a selection of 11 different cryptographic algorithms as key type. For each key type, the online help text provides you with the allowed key length range.

Of particular interest is the *disallowed function* parameter, which lets you specify up to four cryptographic operations for which the generated key cannot be used. For example, if a key is intended solely for the purpose of encrypting data on your system and then exporting it to another system, you could use this facility to disallow the decryption of the data. Please refer to the help text for all the details.

Running the following GENKR command will create a 256-bit AES cryptographic key labeled CRYPTO_AES_PROD_001, available for all cryptographic operations, and store it in the key store CBX180K in library QGPL:

```

GENKR KEYSTORE(QGPL/CBX180K)
RCDLABEL(CRYPTO_AES_PROD_001)
KEYTYPE(*AES)
KEYSIZE(32)
DISALLOW(*NONE)

```

Upon successful completion of the GENKR command, you will receive a message confirming the outcome:

```


```

```

Message ID . . . . . :   CBX0031           Severity . . . . . :
00
Message type . . . . . :   Information

Date sent  . . . . . :   23-02-08           Time sent  . . . . . :
17:22:52

Message . . . . . :   Cryptographic key CRYPTO_AES_PROD_001
successfully
generated.

Cause . . . . . :   A cryptographic key was successfully generated
and stored
under the key record label CRYPTO_AES_PROD_001 in key store
CBX180K in
library QGPL.

```

If you at a later point want to confirm the current presence and status of a cryptographic key, the Retrieve Key Record Attributes (Qc3RetrieveKeyRecordAtr) API comes in handy. As of release V5R4, you need to know the qualified key store name as well as the key record label to get at this information, which I'm making easily accessible with the Display Key Record Attributes (DSPKRA) command:

```

                                Display Key Record Attributes (DSPKRA)

Type choices, press Enter.

Key store  . . . . .                               Name
Library    . . . . .                               *LIBL      Name, *LIBL,
*CURLIB
Record label . . . . .
Output     . . . . .                               *           *, *PRINT

```

If you specify the correct information as input to the command, you'll get a panel similar to the one below in return:

```

                                Display Key Record Attributes

WYNDHAMW                                           23-02-08

17:29:52

```

```

Key Store . . . . . : CBX180K

Library . . . . . : QGPL

Record label . . . . . : CRYPTO_AES_PROD_001

Key type . . . . . : AES

Key size . . . . . : 32

Master key ID . . . . . : 1

Key verification value . . :
9D98E5EC67DC80DF87FBBF481BC46E183A270915

Disallow functions . . . . : *NONE

Press Enter to continue

F3=Exit   F11=Display character KVV   F12=Cancel

F21=Print key record attributes

```

Again, full online and cursor sensitive is provided to ensure that you get all the details needed to document both command and display panel.

At release V6R1, new Cryptographic Services Key Management APIs will be provided to list all key records in a key store (as well as key store attributes), so at that point you will be able to also write a *Work with Key Records* command that shows all a key store's key records, without you having to remember each key record label.

From today's delivery of key record commands, only the Delete Key Record (DLTKR) command remains. Here's what the very simple DLTKR command prompt looks like:

Delete Key Record (DLTKR)

Type choices, press Enter.

Key store	Name
Library	*LIBL Name, *LIBL,
*CURLIB	
Record label	

To delete a key record, simply specify the qualified key store name and the record label of the key record you want to dismiss. But be sure to please consult the command help text before attempting to run this command!

The Delete Key Record API needs *OBJOPR and *DLT authority to the key store containing the key record in order to successfully remove the key store record. To keep the DLTKR command aligned with the other critical and potentially damaging cryptographic CL commands I have provided so far, I've replaced this requirement with function usage authorization. Function usage authorization has been discussed in detail in previous installments of this column, so please refer to the links below for more information on this subject.

Any user attempting to run the DLTKR command will require common object authority to the command as well as be specifically and individually authorized to the CBX_CRYPTO_KEYRECORD_DELETE function usage that is registered by the CBX187M CL program included with this article to build the GENKR, DSPKRA and DLTKR commands. The user running the CBX187M CL program will by default be authorized to the DLTKR command.

Use the following command to locate and change the function usage registrations applying to the key management utilities delivered with the Cryptographic Key Management articles in previous APIs by Example articles:

```
WRKFCNUSG FCNID(CBX_CRYPTO_*)
```

If you've loaded and installed the commands and usage registrations provided with this and the previous APIs by Example articles, you should see a list similar to the one below, following a successful execution of the command above:

Work with Function Usage

Type options, press Enter.

```

2=Change usage    5=Display usage

Opt  Function ID                                Function Name
deletion
      CBX_CRYPT0_KEYRECORD_DELETE              Cryptograhic key record
translation
      CBX_CRYPT0_KEYSTORE_XLATE                Cryptograhic key store
      CBX_CRYPT0_MASTERKEY_CLEAR              Clear cryptograhic master key
load
      CBX_CRYPT0_MASTERKEY_LOAD                Cryptograhic master key part
      CBX_CRYPT0_MASTERKEY_SET                Set cryptograhic master key
      CBX_CRYPT0_MASTERKEY_TEST              Cryptograhic master key test

Bottom
Parameters for option 2 or command

===>

F3=Exit    F4=Prompt    F5=Refresh    F9=Retrieve    F12=Cancel
F17=Top
F18=Bottom

```

Use option 2 to add and/or remove users' function usage. For more information on function usage registration prerequisites and requirements, follow the link to the December 13, 2007, APIs By Example.

While I was creating and testing the commands for today's article, I came across a quite unexpected and peculiar behavior of the Retrieve Key Record Attributes and the Delete Key Record APIs (ILE versions). According to the API documentation both APIs allow the special values *LIBL (Library list) and *CURLIB (Current library) qualification of the key store parameter.

But according to my tests the aforementioned APIs apparently interpret *CURLIB as *LIBL, at least the APIs act functionally equivalent to the behavior expected from the library special value *LIBL, when provided with the library special value *CURLIB.

For some reason it does however not work the other way around. When I specify *LIBL for the library parameter when calling the APIs in question, the APIs return escape message CPF9DB3 Qualified keystore file name is not valid.

To get around this issue, I've included code to resolve the actual key store library name when *LIBL is specified as the key store library qualification. As for the *CURLIB special value, I'm using a feature that might be documented somewhere, but was brought to my attention by Barbara Morris of IBM:

By specifying a replacement value of *CURLIB for the *CURLIB special value in the command definition, the command analyzer will replace the *CURLIB special value with the actual current library of the job running the command. If the job currently has no current library, library QGPL is passed to the command processing program. Here's how it looks in the command source source -- note the replacement value *CURLIB following the special value *CURLIB: That's what does the trick:

```
Q0001:  Qual                      *Name      10
        Min( 1 )
        Expr( *YES )

        Qual                      *Name      10
        Dft( *LIBL )
        SpcVal(( *LIBL )
              ( *CURLIB *CURLIB ))
        Expr( *YES )
        Prompt( 'Library' )
```

As for the funny behavior of the two APIs in question, I'll report my findings to IBM and report back to you what I find out. Finally, let's take a brief look at the CMDCRPSRV menu that I've created to keep all my key management commands in one place -- here's what you get when you run the command GO CMDCRPSRV:

```
CMDCRPSRV                      Cryptographic Services Commands

                                                                    System:
WYNDHAMW
Select one of the following:

Master key commands

    1. Load Master Key Part
LODMSTKP
    2. Set Master Key
SETMSTK
    3. Test Master Key
TSTMSTK
    4. Clear Master Key
CLRMSTK

Key store commands

    11. Create Key Store
```

```

CRTKS
  12. Translate Key Store
TRNKS

Key record commands

  21. Generate Key Record
GENKR
  22. Display Key Record Attributes
DSPKRA
  23. Delete Key Record
DLTKR

More...
Selection or command

===>

F3=Exit   F4=Prompt   F9=Retrieve   F12=Cancel

```

I've already mentioned a couple of V6R1 news in the area of Cryptographic Services APIs, but I thought I'd use this opportunity to go through other exiting new offerings that I've discovered in my initial research when the V6R1 Information Center came online recently.

Release V6R1 offers four new cryptographic services key management exit programs that are triggered by the following four critical key management events, respectively:

- A clear master key operation
- A set master key operation
- A translate key store operation
- A delete key store record operation

The above events could occur either by executing a key management CL command, a key management API or through the Operations Navigator key management GUI interface. Here's an excerpt from the online documentation:

- Clear Master Key (QIBM_QC3_CLR_MSTKEY) is called when the Clear Master Key (CLRMSTKEY) CL command, the Qc3ClearMasterKey API, or the Clear Master Key GUI dialog is being used.
- Delete Keystore Record (QIBM_QC3_DLT_KREC) is called when the Remove Keystore File Entry (RMVCKMKSFE) CL command, the Qc3DeleteKeyRecord API, or the delete action of the Keystore Contents GUI panel is being used.

- Set Master Key (QIBM_QC3_SET_MSTKEY) is called when the Set Master Key (SETMSTKEY) CL command, the Qc3SetMasterKey API, or the set action of the Manage Master Keys GUI panel is being used.
- Translate Keystore (QIBM_QC3_TRN_KSF) is called when the Translate Keystore File (TRNCKMKSF) CL command, the Qc3TranslateKeyStore API, or the Translate Keystore GUI dialog is being used.

Below you'll find an adapted excerpt from the exit point documentation describing the three events that may cause each exit point to be called:

1. Check:
The exit program is being called before the key management event is taking place. The exit program should determine if the event is allowed to take place and perform any necessary pre-processing actions. The exit program should set the Status output parameter value appropriately.
2. Execute:
The exit program is being called after the key management event has taken place and completed successfully. The exit program should perform any necessary post-processing actions.
3. Cancel:
The exit program is being called because another exit program has returned a Status output parameter value indicating the key management event is not allowed to take place. You may need to back out pre-processing actions.

One use of the *Check* event could be to perform the special function usage check that I've added to my suite of key management CL commands, to control access to these commands. The exit program approach allows you to keep such a check in one place and at the same time ensure that the check is always performed, regardless of the interface being used.

Another use of the *Execute* event is to monitor master key settings, and thereby establish an automated master key translation procedure for all key stores being assigned the changing master key.

For further details documenting the Cryptographic Services Exit Programs, follow the link pointing to the Cryptographic Services APIs V6R1 at the end of this article.

As I mentioned, IBM also introduces a set of native i5/OS key management CL commands at release V6R1, deeming my attempt to provide some of these commands to be relevant for release V5R4 only. Hopefully you'll still find my offering useful as templates for your own programming experience with the key management APIs that I'm exploiting in my ditto CL commands. To give you a quick overview of all the new native key management CL commands and provide a short cut to their detailed documentation, I've written up a list of them all at the end of this article, including links to the V6R1 online documentation.

At V6R1 master keys are now included in the Save System (SAVSYS) save operation, and two new special purpose master keys have been added to the master key repository:

1. The Auxiliary Storage Pool (ASP) Master Key, which is used for ASP encryption.

2. The Save/Restore (SAVSYS) Master Key, which is used for encrypting the master keys while on SAVSYS media.

Again, the links below point to more information about *Disk Encryption*, *Backing Up Encrypted Auxiliary Storage Pools*, and *Saving and Restoring Master Keys*.

This APIs by Example includes the following sources:

```
CBX180  -- RPGLE  -- Cryptographic Key Management - Services
CBX180B -- SRVSRC -- Cryptographic Key Management - Binder source

CBX180U -- UIM    -- Cryptographic Services Commands Menu

CBX185  -- RPGLE  -- Create Key Store - CPP (updated version)

CBX187  -- RPGLE  -- Generate Key Record - CPP
CBX187H -- PNLGRP -- Generate Key Record - Help
CBX187X -- CMD    -- Generate Key Record

CBX188  -- RPGLE  -- Display Key Record Attributes - CPP
CBX188E -- RPGLE  -- Display Key Record Attributes - UIM Exit Program
CBX188H -- PNLGRP -- Display Key Record Attributes - Help
CBX188P -- PNLGRP -- Display Key Record Attributes - Panel Group
CBX188V -- RPGLE  -- Display Key Record Attributes - VCP
CBX188X -- CMD    -- Display Key Record Attributes

CBX189  -- RPGLE  -- Delete Key Record - CPP
CBX189H -- PNLGRP -- Delete Key Record - Help
CBX189V -- RPGLE  -- Delete Key Record - VCP
CBX189X -- CMD    -- Delete Key Record

CBX187M -- CLP    -- Cryptographic Key Management IV - Build commands
```

To create all above objects, compile and run CBX187M. As always, you'll also find compilation instructions in the respective source headers. I've included an updated version of the Create Key Store (CRTKS) command's CPP to correct a bug in the earlier version that caused the created key store files' descriptive text to have a maximum length of 10 bytes. Please download and install the updated version of the CBX185 program to fix this issue.

Please note that the two previously published commands Add Function Registration (ADDFCNREG) and Change User Function Usage (CHGUSRFCNU) are prerequisite for the Delete Key Record (DLTKR) command to run successfully -- as well as for the CBX185M program to compile.

You can get the sources for the two aforementioned user function commands with the download made available with my previous APIs by Example article of November 8, 2007 -- just follow the link provided below. Successfully compiling and running the CBX180M CL setup program included with that article is also prerequisite to running the CBX187M setup program included today.

Previously published related articles:

APIs by Example, November 8, 2007: Cryptographic Key Management - Loading and Setting Master Keys:

<http://systeminetwork.com/article/apis-example-cryptographic-key-management-loading-and-setting-master-keys>

APIs by Example, December 13, 2007: Cryptographic Key Management - Testing and Clearing Master Keys:

<http://systeminetwork.com/article/apis-example-cryptographic-key-management-testing-and-clearing-master-keys>

APIs by Example, January 24, 2008: Cryptographic Key Management – Creating and Translating Key Stores:

<http://systeminetwork.com/article/apis-example-cryptographic-key-management-creating-and-translating-key-stores>

IBM documentation:

Educational White Paper: Protecting i5/OS Data with Encryption

http://www-03.ibm.com/servers/enable/site/education/abstracts/efbe_abs.html

i5/OS: Cryptography concepts V5R4:

<http://publib.boulder.ibm.com/infocenter/iserics/v5r4/topic/rzajc/rzajcconcepts.htm>

Cryptographic Services Master Keys V5R4:

<http://publib.boulder.ibm.com/infocenter/iserics/v5r4/topic/apis/qc3MasterKeys.htm>

Cryptographic Services Key Store V5R4:

<http://publib.boulder.ibm.com/infocenter/iserics/v5r4/topic/apis/qc3KeyStore.htm>

i5/OS: Cryptography concepts V6R1:

<http://publib.boulder.ibm.com/infocenter/systems/scope/i5os/topic/rzajc/rzajcconcepts.htm>

Cryptographic Services Key Management V6R1:

<http://publib.boulder.ibm.com/infocenter/systems/scope/i5os/topic/rzajc/rzajckeymgmt.htm>

Disk Encryption V6R1:

<http://publib.boulder.ibm.com/infocenter/systems/scope/i5os/topic/rzaly/rzalyencrypt.htm>

Backing Up Encrypted Auxiliary Storage Pools V6R1:

<http://publib.boulder.ibm.com/infocenter/systems/scope/i5os/topic/rzaiu/rzaiuencryptasp.htm>

Saving and Restoring Master Keys V6R1:

<http://publib.boulder.ibm.com/infocenter/systems/scope/i5os/topic/rzajc/rzajcsavemasterkey.htm>

V6R1 Cryptographic Services Key Management CL Commands:

Add Master Key Part (ADDMSTPART) command:

<http://publib.boulder.ibm.com/infocenter/systems/scope/i5os/topic/cl/addmstpart.htm>

Set Master Key (SETMSTKEY) command:

<http://publib.boulder.ibm.com/infocenter/systems/scope/i5os/topic/cl/setmstkey.htm>

Check Master KVV (CHKMSTKVV) command:

<http://publib.boulder.ibm.com/infocenter/systems/scope/i5os/topic/cl/chkmstkvv.htm>

Clear Master Key (CLRMSTKEY) command:

<http://publib.boulder.ibm.com/infocenter/systems/scope/i5os/topic/cl/clrmstkey.htm>

Add Keystore File Entry (ADDCKMKSFE) command:

<http://publib.boulder.ibm.com/infocenter/systems/scope/i5os/topic/cl/addckmksfe.htm>

Create Keystore File (CRTCKMKSF) command:

<http://publib.boulder.ibm.com/infocenter/systems/scope/i5os/topic/cl/crtckmksf.htm>

Display Keystore File Entry (DSPCKMKSFE) command:

<http://publib.boulder.ibm.com/infocenter/systems/scope/i5os/topic/cl/dspckmksfe.htm>

Generate Keystore File Entry (GENCKMKSFE) command:

<http://publib.boulder.ibm.com/infocenter/systems/scope/i5os/topic/cl/genckmksfe.htm>

Remove Keystore File Entry (RMVCKMKSFE) command:

<http://publib.boulder.ibm.com/infocenter/systems/scope/i5os/topic/cl/rmvckmksfe.htm>

Translate Keystore File (TRNCKMKSF) command:

<http://publib.boulder.ibm.com/infocenter/systems/scope/i5os/topic/cl/trnckmksf.htm>

This article demonstrates the following Cryptographic Services API:

Generate Key Record (Qc3GenKeyRecord) API:

<http://publib.boulder.ibm.com/infocenter/iserics/v5r4/topic/apis/qc3genkr.htm>

Retrieve Key Record Attributes (Qc3RetrieveKeyRecordAtr) API:

<http://publib.boulder.ibm.com/infocenter/iserics/v5r4/topic/apis/qc3rtvka.htm>

Delete Key Record (Qc3DeleteKeyRecord) API:

<http://publib.boulder.ibm.com/infocenter/iserics/v5r4/topic/apis/qc3dltkr.htm>

Key Management APIs V5R4:

<http://publib.boulder.ibm.com/infocenter/iserics/v5r4/topic/apis/catcrypt6.htm>

Cryptographic Services APIs V5R4:

<http://publib.boulder.ibm.com/infocenter/iserics/v5r4/topic/apis/catcrypt.htm>

Key Management APIs V6R1:

<http://publib.boulder.ibm.com/infocenter/systems/scope/i5os/topic/apis/catcrypt6.htm>

Cryptographic Services APIs V6R1 – Including Exit Point Documentation:

<http://publib.boulder.ibm.com/infocenter/systems/scope/i5os/topic/apis/catcrypt.htm>

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

http://www.pentontech.com/IBMContent/Documents/article/56351_470_CrtDspDelKey.zip.

Source URL: <http://iprodeveloper.com/rpg-programming/apis-example-cryptographic-key-management-creating-displaying-and-deleting-key-recor>