


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

APIs by Example: More New Cryptographic Services APIs and Exit Points at Release 6.1

[System iNetwork Programming Tips Newsletter](#)

[Carsten Flensburg](#)

Carsten Flensburg

Thu, 07/22/2010 (All day)

This APIs by Example continues the coverage begun in the preceding installment of this column (see ["New Cryptographic Services APIs and Exit Points at Release 6.1,"](#) June 24, 2010, article ID 65255 at SystemiNetwork.com). That coverage is of the new Cryptographic Services APIs and exit points introduced by IBM with release 6.1. For today's issue, I've created a new Work with Key Store Records (WRKKR) CL command based on the Retrieve Keystore Records (Qc3RetrieveKeyStoreRecords) API, which lets you retrieve a list of all the cryptographic keys stored in a key store. In addition to listing key store records, the WRKKR command provides an interface to other previously published cryptographic key work management commands, letting you display, print, and delete the cryptographic key records and also display the key store file attributes.

Last time, I presented two cryptographic key management exit programs enabled by the cryptographic key management exit points introduced with release 6.1. The exit programs in question are called whenever a master key is set or cleared. For the remaining two exit points covering the events of a key store being translated, meaning all stored keys are encrypted under a new master key, as well as a key record being deleted from a key store, I include in today's issue a couple of new sample exit programs and also information about how to register these exit programs.

As explained last time, the key management exit points are called whenever the associated key management event is taking place, irrespective of the interface used to perform the key management function, being either a CL command, an API, or System i Navigator dialog panels. The translate key store and delete key store record exit programs included today are based on the User Function Registration facility, which I also discussed in last month's API by Example, and it employs the user functions registered as part of earlier installation programs provided. If you've already installed the commands and service program included last month, you should be OK; otherwise please do the installation before registering the exit programs I present this month.

To register the Translate Key Store and Delete Key Store Record exit programs included today, you can either run the two commands below or follow the instructions in the source header of the CBX217M utility install program, to have the install program do it for you.

```
AddExitPgm ExitPnt( QIBM_QC3_TRN_KSF )
              Format( TRNK0100 )
              PgmNbr( *LOW )
              Pgm( /CBX217X1 )
              Text( 'Translate Key Store Exit Program' )

AddExitPgm ExitPnt( QIBM_QC3_DLT_KREC )
              Format( DLKR0100 )
              PgmNbr( *LOW )
```

```
Pgm( /CBX217X2 )
Text( 'Delete Key Store Record Exit Program' )
```

You'll find links to IBM's exit point documentation at the end of this article.

As mentioned last time: To configure the User Function Registration facility once you've installed the utilities included today, run the following command:

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

This command brings up a panel listing all User Function Registration entries applying to the cryptographic key management utilities presented in this and earlier APIs by Examples, and from this panel you can display, add, and remove the user profiles allowed for each function. By default, the user profile running the CL programs installing the various cryptographic key management commands is allowed to perform all functions registered by the install program.

The Qc3RetrieveKeyStoreRecords API has three input parameters specifying the length of the return variable receiving the key store record list, the format of the list, and the qualified name of the key store whose records are returned. The API, in addition to the standard API error data structure, returns its information in two separate return parameters. In the receiver variable, you get an array of list entries, one for each key store record, and the other return parameter holds the list information defining the total length of all list entries available, the number of key records, and the length of each list entry. Armed with this information, you can perform the following programming tasks quite easily:

- Ensure that the receiver variable parameter (a) is capable of holding all the key store record list entries available by allocating a sufficient amount of storage.
- Map the key store record list entry definition in the form of a data structure (b) based on a space pointer (c) to the first entry available, if any.
- Process the key store record data structure for each list entry available (d) and add each record to the key store record list (e).
- Map the key store list entry to the next available list entry by adding the entry length until the number of list entries returned have all been processed (f).
- The above sequence of events is demonstrated in the following code snippet:

```
DoU  RtnRcdFbi.BytAvl <= ApiRcvSiz      Or
      ERRC0100.BytAvl  > *Zero;

(a)  If  RtnRcdFbi.BytAvl > ApiRcvSiz;
      ApiRcvSiz = RtnRcdFbi.BytAvl;
      pRcvVar   = %ReAlloc( pRcvVar: ApiRcvSiz );
      EndIf;

(b)  pKSRA0100 = pRcvVar;

(c)  RtvKsfRcd( KSRA0100
                : ApiRcvSiz
                : RtnRcdFbi
                : 'KSRA0100'
                : PxKsfNam_q
                : ERRC0100
```

```

        );
    EndDo;

    If  ERRC0100.BytAvl > *Zero;
        ExSr  EscApiErr;
    Else;
        ExSr  PrcRcdEnt;
    EndIf;

    BegSr PrcRcdEnt;

(d)    For  Idx = 1  to RtnRcdFbi.NbrKsfRcd;

(e)        ExSr  PutLstEnt;

        If  Idx < RtnRcdFbi.NbrKsfRcd;
(f)        Eval pKSRA0100 += RtnRcdFbi.EntLen;
        EndIf;
    EndFor;

    EndSr;

```

Note that in order to be able to explicitly deallocate the storage allocated for the API receiver variable, I use one space pointer variable for the storage allocation and another one for the key store record list entry processing. This is necessary because the latter is being used for the entry list processing, which implies that the pointer for each returned list entry is being incremented by the list entry length. To successfully deallocate storage, however, you must specify the pointer value originally returned by the storage allocation function, so for that purpose, I retain this value in the pRevVar pointer variable and initially copy it to the pointer variable upon which the key store record data structure is based.

As for the Work with Key Store Record (WRKKR) command, once you've completed the installation of the command and its objects, you'll get the following screen when prompting the command:

Work with Key Store Records (WRKKR)			
Type choices, press Enter.			
Key Store		Name
Library	*LIBL	Name, *LIBL,
*CURLIB			
Output	*	*, *PRINT

You specify the qualified name of a key store and whether the command output should go to a display panel or a printed list. The accompanying help text provides further explanation of the command and its parameters. To give an example of what the Work with Key Store Record (WRKKR) command's work with-panel looks like, I ran the following command on my system:

```
WRKKR  KEYSTORE(QGPL/KEYSTORE01)
        OUTPUT(*)
```

The resultant list is displayed below:

```

                                Work with Key Store Records
WYNDHAMW
11-07-10
12:45:38
Key store . . . . . : KEYSTORE01

Library . . . . . : QGPL

Type options, press Enter.

4=Delete   5=Display   6=Print

                                Master   Modified
Opt  Record label      Key type      Key size   Key      Date and
time
      CBX_WEB_001      *AES          16         1
20100201095812
      CBX_WEB_002      *AES          24         1
20100311181600

Bottom
Parameters or command

===>

F3=Exit      F4=Prompt      F5=Refresh      F6=Generate key record
F9=Retrieve
```

```
F11=View 2    F12=Cancel    F22=Display entire/hex field value
F24=More keys
```

The list panel options include delete, display, and print functions employing the Delete Key Record (DLTKR) and Display Key Record Attributes (DSPKRA) commands, respectively. Function key F6 runs the Generate Key Record (GENKR) command, adding the new key to the key store. Using function key F11, you can toggle between the above list view and an alternative view including the master key verification value and the disallowed cryptographic functions for the key in question.

Function key F20 executes the Display Key Store File Attributes (DSPKSFA) command presented last month, and function key F22 displays the entire record label in case it exceeds the list column size allocated as well as the hexadecimal representation of the master key verification value. The display panel is fully documented in the accompanying help text panel group. Point your cursor to the location of interest and press function key F1 to display the help text applying to that particular area or field.

In addition to the WRKKR command, I've included an updated version of the Cryptographic Services Commands menu previously published. I've added the two new key management commands presented last month and today to the menu, as well as the related command menus pointing you to the native cryptographic CL commands added with release 6.1, the Cryptographic Key Management Commands, and Master Commands, respectively.

This APIs by Example includes the following sources:

```
CBX180U  -- UIM      -- Cryptographic Services Commands Menu

CBX217   -- RPGLE    -- Work with Key Store Records - CPP
CBX217E  -- RPGLE    -- Work with Key Store Records - UIM Exit Program
CBX217H  -- PNLGRP   -- Work with Key Store Records - Help
CBX217P  -- PNLGRP   -- Work with Key Store Records - Panel Group
CBX217V  -- RPGLE    -- Work with Key store Records - VCP
CBX217X  -- CMD      -- Work with Key Store Records

CBX217X1 -- RPGLE    -- Translate Key Store Exit Program
CBX217X2 -- RPGLE    -- Delete Key Store Record Exit Program

CBX217M  -- CLP      -- Work with Key Store Records - Build command
```

To create all these objects, compile and run the CBX217M program, following the instructions in the source header. If you haven't already run the CBX216M installation program included last time, please do so before running CBX217M, and also please observe the instructions stated in the article accompanying the CBX216M program; I've provided a link to this article below.

IBM documentation:

[System i Security Cryptography PDF](#)

[Cryptography concepts](#)

[Cryptographic services key management](#)

[Managing master keys](#)

[Managing cryptographic key store files](#)

Previously published article on SystemiNetwork.com:

[APIs by Example: New Cryptographic Services APIs and Exit Points at Release 6.1](#)

This article demonstrates the following Cryptographic Services APIs and exit points:

[Retrieve Keystore Records \(Qc3RetrieveKeyStoreRecords\) API](#)

[Translate Keystore Exit Point \(QIBM_QC3_TRN_KSF\)](#)

[Delete Keystore Record Exit Point \(QIBM_QC3_DLT_KREC\)](#)

[Cryptographic Services APIs](#)

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

Source URL: <http://iprodeveloper.com/rpg-programming/apis-example-more-new-cryptographic-services-apis-and-exit-points-release-61>