

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

APIs by Example: Crypto Key Management -- Creating Data Key Stores and More

[System iNetwork Programming Tips Newsletter](#)

[Carsten Flensburg](#)

Carsten Flensburg

Thu, 03/27/2008 (All day)

With release V5R4, IBM added significant and comprehensive support of cryptographic key management to the Cryptographic Services APIs. This support included the introduction of key stores and new key management facilities as well as the integration of these valuable additions to the existing collection of cryptographic functions. These new offerings all in all provide the foundation for a substantial improvement in the i5/OS built-in cryptographic environment and security, as well as a higher level of cohesion between the cryptographic APIs and the applications taking advantage of them.

As discussed earlier in this column, the new key store object offers a two-tier key store facility that allows you to store both key encryption keys and data encryption keys encrypted under the master key assigned to the key store in question. Following this scheme, the data keys stored in the key store will be encrypted by the master key only.

Depending on the number of data encryption keys to manage and exchange, and your preferences as far as key separation and storage is concerned, you might at some point require storing your data encryption keys encrypted under a key encryption key (KEK). Once your data encryption keys are encrypted under the KEK, you could store the data key in, for example, a data area, a user space, or a physical data file, the latter demonstrated in IBM's cryptographic sample application presented in the Cryptographic Services APIs manual in the section called "Scenario: Key Management and File Encryption Using the Cryptographic Services APIs" (link provided at the end of this article).

To unify and ease the integration between APIs and application of the data encryption key entity, I enhanced and adapted the use of validation lists as key stores, which I presented in my first APIs by Example article series covering the Cryptographic Services APIs. This work led to the creation of the Create Data Key Store (CRTDTAKS), the Create Data Encryption Key (CRTDTAK), and the Remove Data Encryption Key (RMVDTAK) commands, which I present in today's installment of APIs by Example.

I've discussed the benefits of letting validation lists serve as key stores in previous articles, particularly in the [Cryptographic Services APIs -- Part 3](#) article. The aforementioned article specifically highlights the following validation list properties as valuable in a key store context:

- Validation list entries are accessible only through APIs
- The list entry key data is stored encrypted
- No Coded Character Set Identifier (CCSID) conversion occurs when storing a list entry
- System values control the audit of all list entry access

Due to its nature, a validation list must always, regardless of its use, be carefully secured and access to it monitored on a regular basis. As a consequence, all my validation lists have their public authority set to *EXCLUDE and are audited by the system audit journal QAUDJRN. Also ensure that validation lists are appropriately included in your back-up schedule. Further considerations and measures are naturally due, reflecting the specific use and purpose of any given validation list.

A rich set of validation list APIs exist to access and manipulate validation lists, and as you will see, I'm making full use of these APIs to provide the necessary service functions to the cryptographic commands and programs that I'm going to present in this and the following APIs by Example article. The first command falling into this category is the Create Data Key Store (CRTDTAKS) command. The CRTDTAKS command prompt displays the following parameters:

Create Data Key Store (CRTDTAKS)

Type choices, press Enter.

Data key store		Name
Library	*CURLIB	Name, *CURLIB
Key encryption key store		Name
Library	*CURLIB	Name, *CURLIB
Key encryption key label		
Text 'description'	*BLANK	

You specify the qualified name of the new data key store, the name of an existing key encryption key store, as well as the label identifying a key encryption key as the primary input parameters. The command's online help text explains all the details. The CRTDTAKS command will create a validation list of the specified key store name and put it in the named library or the job's current library, whichever you prefer.

The validation list will be created with public authority *EXCLUDE and be owned by the issuer of the CRTDTAKS command. Following creation object auditing specifying an object audit value of *ALL will automatically be started for the validation list, and its object user defined attribute will be set to DTAK_STORE to allow easy recognition of the specific use of the validation list.

The qualified key encryption key store name will be added to the key store validation list as a special entry, with a special entry ID name, to prevent any attempts to produce a duplicate and allow for a safe retrieval of the information stored in this entry. You also have the option of specifying a brief text describing the key store validation list.

Once you have successfully run the CRTDTAKS command you will therefore end up having a validation list key store available to store data encryption keys and containing the information

necessary to locate and use the key encryption key to encrypt all data encryption keys before storing them in this key store. So how to put data encryption into the data key store? That's exactly what the Create Data Encryption Key (CRTDTAK) does, and here's how the command prompt looks:

Create Data Encryption Key (CRTDTAK)

Type choices, press Enter.

Data key store		Name
Library	*CURLIB	Name, *CURLIB
Data key label		
Key length	16	16, 24, 32
Key bytes 1-8	*GEN	
Key bytes 9-16	*GEN	

You specify the qualified name of the data key store of choice, a data key label uniquely identifying the data encryption key within the key store as well as the desired key length. You have the option of either specifying the encryption key yourself or having the system generate it for you.

The [AES \(Advanced Encryption Standard\)](#) algorithm is used for all key encryption key and data encryption key operations performed by the CRTDTAK command. The encryption algorithm settings applied to the cryptographic key operations, such as block length, encryption mode, padding and pad characters, are defined by the GetMgtAlg() function in the CBX190 service program. The algorithm settings defined by the GetMgtAlg() function can of course be made subject to changes, as dictated by individual requirements.

At the point where the data key store is created only the existence of the specified key encryption key store is verified, not the key encryption key label. The latter check is however performed when the CRTDTAK command is run, because at this point the key encryption key is actually needed to perform the required encryption of the data encryption key. So for the CRTDTAK command to be able to store a data encryption key in the specified key store, the following events must have taken place:

1. A master key must have been successfully loaded and set, using the Load Master Key Parts (LODMSTKP) and Set Master Key (SETMSTK) commands.
2. A key store must have been created, using the Create Key Store (CRTKS) command and specifying the master key created above as the key store assigned master key.
3. A key encryption key (KEK) must have been named, created and added to the key store created above, using the Generate Key Record (GENKR) command.

4. A data key store must have been created, using the Create Data Key Store (CRTDTAKS) command and specifying the above key encryption key store and key encryption key label as command input parameters.
5. Once the above prerequisite actions have been completed, you can run the Create Data Encryption Key (CRTDTAK) command, specifying the above data key store, a uniquely named data key label as well as the encryption key length and key value as command input parameters to create a data encryption key.

You are now ready to use the created data encryption key to encrypt and decrypt data on your system. An example of how to accomplish that will be the topic for the next installment of APIs by Example.

Last item on today's agenda is the Remove Data Encryption Key (RMVDTAK) command. A cryptographic rule of thumb states to always destroy unused or obsolete encryption keys. For this purpose I have created the RMVDTAK command, which has the following quite predictable appearance:

Remove Data Encryption Key (RMVDTAK)

Type choices, press Enter.

Data key store Name

Library *CURLIB Name, *CURLIB

Key label

The qualified key store name combined with the key label identifies the key to remove from the data key store. The command is executed without further warning once the correct data has been provided and Enter pressed, so due caution is imperative when working with this command or managing access to it. Again, full online and cursor sensitive is provided to ensure that you get all the details documenting the command and its proper use.

To successfully run the three data key CL commands presented today, you'll need a special function usage registration for all user profiles requiring access to these commands. The CBX190M CL program included with this article to create all commands will automatically register the *CBX_CRYPTO_KEY_MANAGEMENT* special function usage and authorize the user profile running the CL program to this function.

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 this and previous APIs by Example articles:

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

Given that you've successfully 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
    CBX_CRYPT0_KEY_MANAGEMENT                    Cryptographic key management
    CBX_CRYPT0_KEYRECORD_DELETE                  Cryptographic key record
deletion
    CBX_CRYPT0_KEYSTORE_XLATE                    Cryptographic key store
translation
    CBX_CRYPT0_MASTERKEY_CLEAR                   Clear cryptographic master key
load
    CBX_CRYPT0_MASTERKEY_LOAD                    Cryptographic master key part
    CBX_CRYPT0_MASTERKEY_SET                     Set cryptographic master key
    CBX_CRYPT0_MASTERKEY_TEST                    Cryptographic 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. As mentioned above there's more information on function usage registration prerequisites and requirements in an earlier APIs by Example article, the one of December 13, 2007, please follow the link provided below to look up that article.

This APIs by Example includes the following sources:

```

CBX190    -- RPGLE    -- Encrypt and Decrypt Data - services
CBX190B   -- SRVSR   -- Encrypt and Decrypt Data - binder source
CBX191    -- RPGLE    -- Cryptographic Data Key Management - services

```

```

CBX191B  -- SRVSRC  -- Cryptographic Data Key Management - binder
source

CBX1910  -- CLLE    -- Create Data Key Store - CPP
CBX1910H -- PNLGRP  -- Create Data Key Store - Help
CBX1910V -- CLLE    -- Create Data Key Store - VCP
CBX1910X -- CMD     -- Create Data Key Store

CBX1911  -- RPGLE   -- Create Data Encryption Key - CPP
CBX1911H -- PNLGRP  -- Create Data Encryption Key - Help
CBX1911V -- RPGLE   -- Create Data Encryption Key - VCP
CBX1911X -- CMD     -- Create Data Encryption Key

CBX1912  -- RPGLE   -- Remove Data Encryption Key - CPP
CBX1912H -- PNLGRP  -- Remove Data Encryption Key - Help
CBX1912V -- RPGLE   -- Remove Data Encryption Key - VCP
CBX1912X -- CMD     -- Remove Data Encryption Key

CBX190M  -- CLP     -- Cryptographic Data Key Management - build
commands

```

To create all above objects, compile and run CBX190M, following the instructions in the source header. As always, you'll also find compilation instructions in the respective source headers.

Please note that prior to compiling (and running) the CBX190M CL program, you'll need to create the Change Object Attribute (CHGOBJATR) command that came with the APIs by Example article of May 12, 2005:

```

CBX135   -- RPGLE   -- Change Object Attributes - CPP
CBX135H  -- PNLGRP  -- Change Object Attributes - Help
CBX135O  -- RPGLE   -- Change Object Attributes - POP
CBX135V  -- RPGLE   -- Change Object Attributes - VCP
CBX135X  -- CMD     -- Change Object Attributes

CBX135M  -- CLP     -- Change Object Attributes - build command

```

For your convenience I've included all sources needed to create the CHGOBJATR command, as well as the CBX135M CL program that will do it for you, with this article. Just follow the instructions in the CBX135M source header.

Please note that the two previously published commands Add Function Registration (ADDFCNREG) and Change User Function Usage (CHGUSRFCNU) are prerequisite for the CBX190M 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 CBX190M setup program included today.

Previously published related articles:

APIs By Example, May 12, 2005:

Change Object Attributes Command:

<http://systeminetwork.com/article/apis-example-change-object-attributes-command>

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>

APIS by Example, February 28, 2008:

Cryptographic Key Management – Creating, Displaying, and Deleting Key Records:

<http://systeminetwork.com/article/apis-example-cryptographic-key-management-%E2%80%93-creating-displaying-and-deleting-key-records>

IBM documentation:

Scenario: Key Management and File Encryption Using the Cryptographic Services APIs:

<http://publib.boulder.ibm.com/infocenter/iseriess/v5r4/topic/apis/qc3Scenario.htm>

Educational White Paper: Protecting i5/OS Data with Encryption:

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

IBM System i Security: Protecting i5/OS Data with Encryption:

<http://www.redbooks.ibm.com/Redbooks.nsf/RedpieceAbstracts/sg247399.html?Open>

This article demonstrates the following Cryptographic Services API:

Encrypt Data (Qc3EncryptData) API:

<http://publib.boulder.ibm.com/infocenter/iseriess/v5r4/topic/apis/qc3encdt.htm>

Decrypt Data (Qc3DecryptData) API:

<http://publib.boulder.ibm.com/infocenter/iseriess/v5r4/topic/apis/qc3decdt.htm>

Translate Data (Qc3TranslateData) API:

<http://publib.boulder.ibm.com/infocenter/iseriess/v5r4/topic/apis/qc3trndt.htm>

Generate Symmetric Key (Qc3GenSymmetricKey) API:

<http://publib.boulder.ibm.com/infocenter/iseriess/v5r4/topic/apis/qc3gensk.htm>

Generate Pseudorandom Numbers (Qc3GenPRNs) API:

<http://publib.boulder.ibm.com/infocenter/iseriess/v5r4/topic/apis/qc3genprns.htm>

Create Algorithm Context (Qc3CreateAlgorithmContext) API:

<http://publib.boulder.ibm.com/infocenter/iseriess/v5r4/topic/apis/qc3crtax.htm>

Create Key Context (Qc3CreateKeyContext) API:

<http://publib.boulder.ibm.com/infocenter/iserics/v5r4/topic/apis/qc3crtkx.htm>

Destroy Algorithm Context (Qc3DestroyAlgorithmContext) API:

<http://publib.boulder.ibm.com/infocenter/iserics/v5r4/topic/apis/qc3desax.htm>

Destroy Key Context (Qc3DestroyKeyContext) API:

<http://publib.boulder.ibm.com/infocenter/iserics/v5r4/topic/apis/qc3deskx.htm>

Retrieve Key Record Attributes (Qc3RetrieveKeyRecordAtr) API:

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

Validation List APIs V5R4:

<http://publib.boulder.ibm.com/infocenter/iserics/v5r4/topic/apis/sec6.htm>

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

http://www.pentontech.com/IBMContent/Documents/article/56462_537_CrtDtaK.zip.

Source URL: <http://iprodeveloper.com/rpg-programming/apis-example-crypto-key-management-creating-data-key-stores-and-more>