

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

APIs by Example: Cryptographic Services APIs, Part 7

[System iNetwork Programming Tips Newsletter](#)

[Carsten Flensburg](#)

Carsten Flensburg

Thu, 03/09/2006 (All day)

This is the final installment of the Cryptographic Services APIs article series. Today, I add the Change Master Key (CHGMSTK) command to the set of cryptographic key management commands and functions that I have presented so far. Changing a master key (or any other cryptographic key) is necessary in the event that the key has been compromised, or as part of a key expiration scheme.

The latter is calculated primarily based on the correlation between the key length in bits and the processor power required to run a successful brute-force attack against the key. One way to protect the key is to change a cryptographic key before it is practically possible to break it. This is only one strategy, however, and it should be combined with other defense lines to ensure an overall sufficient level of security in all cryptographic applications and setups.

For more details about the key length aspect of cryptography, check out the following report:

<http://www.crypto.com/papers/keylength.pdf>

Though the CHGMSTK command on the surface looks exactly like the Create Master Key (CRTMSTK) command, it is faced with a challenge that CRTMSTK is not: One or more key encrypting keys (KEK) could already be encrypted under the current master key.

The CHGMSTK command must therefore be able to retrieve and decrypt any KEK using the current master key, encrypt it using the new master key and update the key store with the changed master key value. Fortunately a Cryptographic Services API that supports that exact requirement exists: The Translate Data (Qc3TranslateData) API takes a data string encrypted under one key and translates that data to encryption under another key -- all in one process.

Because both the decryption and encryption process is performed in one step, the encrypted data is never exposed in its cleartext form. The Translate Data API requires two pairs of context tokens: the decryption key and algorithm context tokens and the encryption key and algorithm context tokens. To get all the details about the Translate Data API, follow the link at the end of this article.

Here's an overview of the Change Master Key process, in steps:

1. Place a lock on the key store validation list to prevent the CHGMSTK command from being run in parallel.
2. Create a key and algorithm context token for the current master key.
3. Change the master key value in the key store as specified on the key input parameter.
4. Create a key and algorithm context token for the new master key.
5. Retrieve a list of all KEKs encrypted under the old master key.

6. For each KEK entry, use the Translate Data API to translate encryption from the old master key to the new master key, using the context tokens from steps 2 and 4.
7. Destroy all context tokens and release the key store lock.

To avoid a potential disaster if the process is interrupted because of an unforeseen event, I strongly recommend that you have a secure and timely backup procedure in place for the key store validation list. A restore could be the only way to recover the key store entries -- and thereby access to the encrypted data -- in such a case.

The CHGMSTK command displays the following prompt:

```

                                Change Master Key (CHGMSTK)
Type choices, press Enter.
Key length . . . . . 16                16, 24, 32
Key bytes 1-8 . . . . . *GEN
Key bytes 9-16 . . . . . *GEN

```

As with all the other key management commands that I've presented so far, special authorization is required to the CBX_CRYPTO_KEY_USAGE user function, which was installed during creation of command objects in a previous article in this series. The command WRKFCNUSG FCNID (CBX_CRYPTO_KEY_USAGE) shows you exactly which user profiles are authorized and also lets you add or remove user profiles.

To verify a successful update of the master key, use the following procedure, which is similar to the test scheme in part six of this article series:

1. Create a master key using the CRTMSTK command: CRTMSTK
2. Create a KEK using the Create Key Encrypting Key (CRTKEK) command: CRTKEK
KEYLABEL(CBX_KEK_0001)
3. Create a data encryption key using the Create Data Encryption Key (CRTDTAK) command:
CRTDTAK KEYLABEL(CBX_DTAK_0001) KEKLABEL(CBX_KEK_0001)
4. Use Data File Utility (DFU) or a similar utility to update the CBX1501F control file. Specify the data key label (in the preceding example, CBX_DTAK_0001) in the KEYLBL field. Specify whichever number you want to be the initial customer number in the field LSTCUS.
5. Run the command ADDCUSRCD to create a customer record.
6. Run the command RUNQRY *N CBX1502F to verify that the record has been added and the customer data encrypted.
7. Run the command CHGCUSRCD, specifying the customer number returned in step 5. You should now be able to see in cleartext the data previously entered.
8. Create a save file and save the key store validation list to that save file:

```

CRTSAVF FILE(/CBX147S)
SAVOBJ OBJ(CBX147L) LIB() DEV(*SAVF) SAVF(CBX147S)

```

9. Change the master key using the CHGMSTK command, allowing it to generate the new key value by defaulting the command: CHGMSTK
10. Repeat step 7. If you can still see the entered data in cleartext, the master key change was successful. Remember to decide what to do with the save file from step 8.

Given the requirement, the CHGMSTK command could of course quite easily be cloned to support the change of other key types. I leave that as an exercise for the reader, however.

In V5R4, IBM has made a new set of Key Management APIs available, which offer great flexibility and built-in security as well as support advanced key management concepts. There is of course a price tag in the form of the challenges and complexity related both to understanding the concepts and to programming the APIs, but for V5R4 and later -- and for the reasons I've mentioned -- I still recommend and encourage the use of IBM's new offering in this area.

The discussions and code in this article series are intended to give you an adequate general introduction to the Cryptographic Services APIs as well as useful API examples to help you take your own next steps in building cryptographic applications. I emphasize that careful and individual consideration should be given to each specific cryptographic task or project to avoid compromising the very data security and privacy that you pursue.

I've provided the following cryptographic and key management functions with this and previous articles in this series:

```

GenAesKey() -- Generate AES cipher key
GenInzVct() -- Generate initialization vector
GetAlgCtx() -- Get algorithm context
GetMgtAlg() -- Get key management algorithm context
GetKeyCtx() -- Get key context
RmvAlgCtx() -- Remove algorithm context
RmvKeyCtx() -- Remove key context
EncDtaStr() -- Encrypt data string using context tokens
DecCphStr() -- Decrypt cipher string using context tokens
TrnDtaStr() -- Translate data string
ChgKeyEnc() -- Change key encryption

AddKeyEnt() -- Add key entry to key store
ChgKeyEnt() -- Change key store entry
ChkSubKey() -- Check sub key existence
FndNxtKeyE() -- Find next key entry
FndTopKeyE() -- Find top key entry
GetKeyAtr() -- Get key attribute
GetKeySto() -- Get key store
GetMstKeyLb() -- Get master key label
RmvKeyEnt() -- Remove key store entry
VfyKeyEnt() -- Verify key store entry
GetFcnUsg() -- Get function usage
GetMstKeyTk() -- Get master key context token
GetKekTkn() -- Get key encryption key context token
GetDtaKeyTk() -- Get data key context token
RmvParCtxTk() -- Remove parent context tokens
GetCphKey() -- Get cipher key

```

NOTE: I recommend reading all parts of this article, and taking into account all recommendations and warnings stated in each part of this article, before using any or part of the tools provided in this article series in a production environment.

You can find part one of this article here:

<http://www2.systeminetwork.com/article.cfm?id=51236>

Part two here:

<http://www2.systeminetwork.com/article.cfm?id=51786>

Part three here:

<http://www2.systeminetwork.com/article.cfm?id=51863>

Part four here:

<http://www2.systeminetwork.com/article.cfm?id=51962>

Part five here:

<http://www2.systeminetwork.com/article.cfm?id=52017>

Part six here:

<http://www2.systeminetwork.com/article.cfm?id=52119>

This APIs by Example includes the following source members:

```
CBX146    -- Cryptographic services service program
CBX146B   -- Service program binder source
CBX147    -- Cryptographic key management service program
CBX147B   -- Service program binder source
```

These sources are all revised versions of previously published sources, which have been updated to support new functions introduced in this article. Please replace these sources in your utility library's source files. The CBX151M program ensures that the service program gets correctly recompiled.

The CHGMSTK command includes the following sources:

```
CBX1511 -- Change Master Key - CPP 1
CBX1512 -- Change Master Key - CPP 2
CBX151H -- Change Master Key - help
CBX151V -- Change Master Key - validity checker
CBX151X -- Change Master Key - command
```

I have included a program that performs all necessary command and object (re)creation:

```
CBX151M -- Command objects creation
```

Compilation instructions are also in the source headers, as usual.

V5R4 Key Management APIs:

<http://publib.boulder.ibm.com/infocenter/iseres/v5r4/topic/apis/cacrypt.htm>

This article demonstrates the following APIs:

Add Validation List Entry (QsyAddValidationLstEntry) API:

<http://publib.boulder.ibm.com/infocenter/iserics/v5r3/topic/apis/qsyavle.htm>

Change Validation List Entry (QsyChangeValidationLstEntry) API:

<http://publib.boulder.ibm.com/infocenter/iserics/v5r3/topic/apis/QSYCVLE.htm>

Find First Validation List Entry (QsyFindFirstValidationLstEntry) API:

<http://publib.boulder.ibm.com/infocenter/iserics/v5r3/topic/apis/QSYFFVLE.htm>

Find Next Validation List Entry (QsyFindNextValidationLstEntry) API:

<http://publib.boulder.ibm.com/infocenter/iserics/v5r3/topic/apis/QSYFNVLE.htm>

Find Validation List Entry (QsyFindValidationLstEntry) API:

<http://publib.boulder.ibm.com/infocenter/iserics/v5r3/topic/apis/QSYFIVLE.htm>

Remove Validation List Entry (QsyRemoveValidationLstEntry) API:

<http://publib.boulder.ibm.com/infocenter/iserics/v5r3/topic/apis/QSYRVLE.htm>

Encrypt data (Qc3EncryptData) API:

<http://publib.boulder.ibm.com/infocenter/iserics/v5r3/topic/apis/qc3encdt.htm>

Decrypt data (Qc3DecryptData) API:

<http://publib.boulder.ibm.com/infocenter/iserics/v5r3/topic/apis/qc3decdt.htm>

Translate Data (Qc3TranslateData) API:

<http://publib.boulder.ibm.com/infocenter/iserics/v5r3/topic/apis/qc3trndt.htm>

Generate Symmetric Key (Qc3GenSymmetricKey) API:

<http://publib.boulder.ibm.com/infocenter/iserics/v5r3/topic/apis/qc3gensk.htm>

Generate Pseudorandom Numbers (Qc3GenPRNs) API:

<http://publib.boulder.ibm.com/infocenter/iserics/v5r3/topic/apis/qc3genprns.htm>

Create Algorithm Context (Qc3CreateAlgorithmContext) API:

<http://publib.boulder.ibm.com/infocenter/iserics/v5r3/topic/apis/qc3crtax.htm>

Create Key Context (Qc3CreateKeyContext) API:

<http://publib.boulder.ibm.com/infocenter/iserics/v5r3/topic/apis/qc3crtkx.htm>

Destroy Algorithm Context (Qc3DestroyAlgorithmContext) API:

<http://publib.boulder.ibm.com/infocenter/iserics/v5r3/topic/apis/qc3desax.htm>

Destroy Key Context (Qc3DestroyKeyContext) API:

<http://publib.boulder.ibm.com/infocenter/iserics/v5r3/topic/apis/qc3deskx.htm>

Send Program Message (QMHSNDPM) API:

<http://publib.boulder.ibm.com/infocenter/iserics/v5r3/topic/apis/QMHSNDPM.htm>

Move Program Messages (QMHMOVPM) API:

<http://publib.boulder.ibm.com/infocenter/iserics/v5r3/topic/apis/qmhmovpm.htm>

Resend Escape Message (QMHRSNEM) API:

<http://publib.boulder.ibm.com/infocenter/iserics/v5r3/topic/apis/QMHRSNEM.htm>

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

http://www.pentontech.com/IBMContent/Documents/article/52224_59_CryptoServices7.zip

Source URL: <http://iprodeveloper.com/rpg-programming/apis-example-cryptographic-services-apis-part-7>