

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

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

[System iNetwork Programming Tips Newsletter](#)

[Carsten Flensburg](#)

Carsten Flensburg

Thu, 06/24/2010 (All day)

Today I will take a look at some of the Cryptographic Services APIs and Exit Points that were introduced with release 6.1. In previous articles I have been covering the Cryptographic Services APIs up to and including release 5.4. With release 6.1, IBM continued enhancing and adding new APIs to the cryptographic API sets in continuation of the significant changes included with each release since the first limited supply of cryptographic APIs were PTF'ed to release 5.2 back in 2003. As such, I decided to continue my coverage of this current and interesting topic. Based on the 6.1 changes, I've updated a number of previously published key management CL commands and also included a new CL command displaying a key store file's attributes.

Cryptographic key management has been the focus area of the most recent updates and enhancements added to the cryptographic APIs. In extension to the Key Management API offering, IBM has also introduced Key Management CL commands, as well as corresponding System i Navigator facilities. Given this variety of key management maintenance interfaces, IBM also added also added four new exit points with release 6.1, letting you control access to critical maintenance activities beyond the levels enabled by user profile, special authority, and resource authority. I have included a couple of exit program examples with this article for you to use as a starting point in case you decide to take advantage of these new exit points.

In a couple of APIs by Example articles following the release 5.4 key management API additions, I presented four master key CL commands based on functionally corresponding APIs. While IBM with release 6.1 not only added two new system master keys for ASP and Save/Restore encryption purposes, respectively, but also had the courtesy of adding master key CL commands, although named differently, but otherwise similar to my earlier contribution in this area, I however still decided to update my master key commands to reflect the new system master key enhancements.

Partly driven by the affection that comes from the attachment to your own creations, but also based on the observation that the release 5.4 key management commands offer a number of output options, not included with IBM's versions, I found this effort worthwhile and have therefore added the master key special values \*ASP and \*SAVRST to the following master key commands:

- Load Master Key Part (LODMSTKP)
- Set Master Key (SETMSTK)
- Test Master Key (TSTMSTK)
- Clear Master Key (CLRMSTK)

I've updated the commands' help text and message descriptions to also reflect the aforementioned change. At the end of this article you'll find a source list naming the relevant source members as well as instructions on how to perform the command updates. In case you're interested in learning more

about the native master key commands that IBM included with release 6.1, please follow the links below taking you to the documentation of the following CL commands:

- Add Master Key Part (ADDMSTPART)
- Set Master Key (SETMSTKEY)
- Check Master KVV (CHKMSTKVV)
- Clear Master Key (CLRMSTKEY)

As mentioned earlier new exit points have been made available to accommodate the need for tighter security around the key management maintenance interfaces. Any of the above CL commands and related APIs capable of performing any kind of master key update, all require user profile special authority \*ALLOBJ as well as \*SECADM.

When I wrote my key management commands I further added an authorization layer based on the User Function Registration concept, which allows you to control access within your applications and modules at a more granular level than other common authorization sources. For a detailed discussion of this concept, including potential pitfalls and shortcomings please lookup the article series covering this topic; you'll find links at the end of this article. Anyway, with the introduction of the new key management exit points I've integrated the User Function Registration approach directly into the exit program.

Consequently, whenever a user performs either a set master key or clear master key operation, irrespective of the interface used, the exit point will call the registered exit program, which in turn will interrogate the User Function Registration facility to verify that the user profile in question is allowed to proceed, before the requested action is actually carried out.

Since more exit programs can be registered for a single exit point and each exit program has the option of cancelling the operation, the exit point not only calls the registered exit program prior to the operation, but also following either a successful or failed operation. This enables you to securely perform post-processing or back-out processing activities, respectively, if applicable.

I've included links to the complete documentation of these exit points below as well as in the sample exit program source headers, and provide a couple of sample exit programs implementing the two exit point interfaces and the User Function Registration based access control as outlined above. Once the exit programs have been created, you'd want to run the following commands to activate the exit programs for the set and clear master key events:

```
AddExitPgm ExitPnt( QIBM_QC3_SET_MSTKEY )
              Format( STMK0100 )
              PgmNbr( *LOW )
              Pgm( /CBX216X1 )
              Text( 'Set Master Key Exit Program' )

AddExitPgm ExitPnt( QIBM_QC3_CLR_MSTKEY )
              Format( CLMK0100 )
              PgmNbr( *LOW )
              Pgm( /CBX216X2 )
              Text( 'Clear Master Key Exit Program' )
```

To configure the User Function Registration facility once you have installed the utilities included today, run the following command:

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

This will bring 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.

In previous APIs by Examples articles I've discussed the significance and implications of cryptographic key management in great detail as well as provided key management solutions addressing the restrictions and limitations imposed on earlier releases. So in case you're still stuck on release 5.3 or earlier, you might find these earlier key management utilities useful. I've included links below to one of these API by Examples articles, as well as more recent ones taking advantage of the release 5.4 key management API offerings in this area. In the latter articles you'll also find a detailed explanation of the key store object and how it is employed in a modern cryptographic infrastructure. In the event that you'd like to learn more about key store concept and object I suggest you look up these articles and dive into the details.

For the benefit of those requiring just a brief explanation, here goes: Key stores solve the cryptographic challenge of storing cryptographic keys encrypted under a master key in order to protect the cryptographic key from exposure to unauthorized parties. The cryptographic keys are identified by a key label, which is used to reference the cryptographic key on input and output operations to the key store.

Basically key stores are implemented as file objects. The object as such has no explicit attributes defining it is a key store. The file is however defined in such a way that only system functions are allowed to do input and output operations against the file. Any other attempt to access the key store content will fail. In essence this implies that only the Cryptographic Services APIs and internal system functions are allowed to access the key records stored in the key store.

The following Cryptographic Services APIs are available to manage key stores and key store records:

- Create Keystore (Qc3CreateKeyStore) creates a database file for securely storing cryptographic key values.
- Retrieve Keystore File Attributes (Qc3RetrieveKeyStoreFileAtr) returns for the specified key store file key store file specific attributes.
- Translate Keystore (Qc3TranslateKeyStore) translates keys stored in the specified key store files to another master key, or to the current version of the master key.
- Delete Key Record (Qc3DeleteKeyRecord) deletes a key record from a key store file.
- Generate Key Record (Qc3GenKeyRecord) generates a random key or key pair and securely stores it in a key store file.
- Retrieve Key Record Attributes (Qc3RetrieveKeyRecordAtr) returns the key type and key size of a key stored in a key store file.
- Retrieve Keystore Records (Qc3RetrieveKeyStoreRecords) returns a list of key store records and their attributes for a key store file.

- Write Key Record (Qc3WriteKeyRecord) securely stores the specified key value in a key store file.

There's no Delete Keystore API, so when you want to delete a key store file you must use the Delete File (DLTF) command. Many of the above APIs have been subject to discussion and API code examples in earlier articles in this newsletter. Again, you'll find links to these articles at the end of this one. The Qc3RetrieveKeyStoreFileAttr API is however a release 6.1 invention and has therefore not been presented before. So today I'll show you an example of how to code this API and make the retrieved key store file information available.

The Qc3RetrieveKeyStoreFileAttr API parameter list is very similar to numerous other retrieve APIs, so there's no surprise waiting for you here. There is a total of five required parameters, including a receiver variable, a receiver variable length, the receiver variable format, the qualified name of the key store and finally the API standard error code data structure. The receiver variable is defined as a data structure, which apart from the bytes returned and bytes available subfields will give you the following four key store file attributes:

- The number of key records contained in the key store file
- The master key ID of the master key used to encrypt the key store records
- The date and time the key store file was last translated (or created)
- The translation status of the key records stored in the key store

As for the translation status attribute, this indicates whether the key store records are stored under the current version of the master key protecting the key store, or the key records are stored under the old (previous) version of the master key, or if the key records are stored under a version of the master key no longer known to the system. The latter situation will require a restore of the master key in order to be able to retrieve the key records encrypted under this lost version of the master key.

If you rely on key stores in your cryptographic infrastructure and business data is encrypted using key records stored in these key stores, it is evident that careful monitoring and management of your key stores and key records is absolutely critical and should be a mandatory part of your business critical security procedures. For the purpose of easily accessing or documenting the key store file attributes of a key store, the Display Key Store File Attr (DSPKSFA) command provides a useful solution:

```

Display Key Store File Attr (DSPKSFA)

Type choices, press Enter.

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

```

Simply specify the qualified name of the key store and whether the command output should go to a display panel or printed list. Help text is also provided to further explain the command and its parameters. To give an example of what the command output looks like, I ran the following command on my system:

```
DSPKSFA  KEYSTORE (KEYSTORE01)
          OUTPUT (*)
```

The above command produced the display panel shown below:

```

                                Display Key Store File Attributes
WYNDHAMW
                                17-06-10
19:00:31
Key store file . . . . :  KEYSTORE01

Library . . . . . :  QGPL

Master key . . . . . :  2

Number of key records . . . . . :  7

Translation date and time . . . . :  16-06-2010  18:20:22

Translation status . . . . . :  *CURRENT

Text 'description' . . . . . :  Key store 001

Bottom
F3=Exit  F5=Refresh  F8=Work with key store object  F12=Cancel

```

You'll note that in addition to the mentioned key store file attributes I've added the textual description of the file object to the information displayed. The display panel is documented in the online and cursor sensitive 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. To print the information either use the Print function key in the above display panel or specify OUTPUT(\*PRINT) on the DSPKSFA command prompt. To work with the key store file object you can use function key F8 in the above display panel.

**This APIs by Example includes the following sources:**

```

CBX180    -- RPGLE    -- Cryptographic Key Management - Services
CBX180B   -- SRVSRV   -- Cryptographic Key Management - Binder source

CBX181    -- RPGLE    -- Load Master Key Part - CPP
CBX181H   -- PNLGRP   -- Load Master Key Part - Help
CBX181X   -- CMD      -- Load Master Key Part

CBX182    -- RPGLE    -- Set Master Key - CPP
CBX182E   -- RPGLE    -- Set Master Key - UIM Exit Program
CBX182H   -- PNLGRP   -- Set Master Key - Help
CBX182P   -- PNLGRP   -- Set Master Key - Panel Group
CBX182V   -- RPGLE    -- Set Master Key - VCP
CBX182X   -- CMD      -- Set Master Key

CBX183    -- RPGLE    -- Test Master Key - CPP
CBX183E   -- RPGLE    -- Test Master Key - UIM Exit Program
CBX183H   -- PNLGRP   -- Test Master Key - Help
CBX183P   -- PNLGRP   -- Test Master Key - Panel Group
CBX183V   -- RPGLE    -- Test Master Key - VCP
CBX183X   -- CMD      -- Test Master Key

CBX184    -- RPGLE    -- Clear Master Key - CPP
CBX184H   -- PNLGRP   -- Clear Master Key - Help
CBX184X   -- CMD      -- Clear Master Key

CBX216    -- RPGLE    -- Display Key Store File Attributes - CPP
CBX216H   -- PNLGRP   -- Display Key Store File Attributes - Help
CBX216P   -- PNLGRP   -- Display Key Store File Attributes - Panel Group
CBX216V   -- RPGLE    -- Display Key Store File Attributes - VCP
CBX216X   -- CMD      -- Display Key Store File Attributes

CBX216X1  -- RPGLE    -- Set Master Key Exit Program
CBX216X2  -- RPGLE    -- Clear Master Key Exit Program

CBX180M   -- CLP      -- Cryptographic Key Management - Build commands
CBX183M   -- CLP      -- Cryptographic Key Management II - Build commands

```

```
CBX216M  -- CLP      -- Display Key Store File Attributes - Build
command
```

To create all the above cryptographic commands objects, compile the CBX183M program and then compile and run the CBX180M program, following the instructions in the source headers. Note that you'll need to download and install the User Function Usage commands published in the August 25, 2005 issue of APIs by Example; Add Function Registration (ADDFCNREG) and Change User Function Usage (CHGUSRFCNU). Link to the article is included below. You'll find additional compilation instructions in the respective source headers.

Finally, compile and run the CBX216M program, again following the instructions in the source header. Specifically note that if you want the CBX216M program to register the key management exit programs, you'll need to update a program variable prior to compiling the program. Again, see the source header for more details.

### **IBM Documentation:**

[System i Security Cryptography PDF](#)

[Cryptography concepts](#)

[Cryptographic services key management](#)

[Managing master keys](#)

[Managing cryptographic key store files](#)

### **Master key CL commands:**

[Add Master Key Part \(ADDMSTPART\)](#)

[Set Master Key \(SETMSTKEY\)](#)

[Check Master KVV \(CHKMSTKVV\)](#)

[Clear Master Key \(CLRMSTKEY\)](#)

### **Related articles:**

[APIs by Example: Crypto Key Management -- Creating Data Key Stores and More](#)

[APIs by Example: Cryptographic Key Management - Loading and Setting Master Keys](#)

[APIs by Example: Cryptographic Key Management - Testing and Clearing Master Keys](#)

[APIs by Example: Cryptographic Key Management - Creating and Translating Key Stores](#)

[APIs by Example: Cryptographic Key Management -- Creating, Displaying, and Deleting Key Records](#)

[APIs by Example: Crypto Key Management -- Creating Data Key Stores and More](#)

[APIs by Example: Crypto Key Mgmt-Encrypt/Decrypt with Key Hierarchy](#)

[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](#)

**This article demonstrates the following Cryptographic Services APIs and Exit Points:**

[Retrieve Keystore File Attributes \(Qc3RetrieveKeyStoreFileAtr\) API](#)

[Load Master Key Part \(Qc3LoadMasterKeyPart\) API](#)

[Set Master Key \(Qc3SetMasterKey\) API](#)

[Test Master Key \(Qc3TestMasterKey\) API](#)

[Clear Master Key \(Qc3ClearMasterKey\) API](#)

[Set Master Key Exit Point \(QIBM QC3 SET MSTKEY\)](#)

[Clear Master Key Exit Point \(QIBM QC3 CLR MSTKEY\)](#)

[Cryptographic Services APIs](#)

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

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