

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

## APIs by Example: Cryptographic Services APIs

[System iNetwork Programming Tips Newsletter](#)

[Carsten Flensburg](#)

Carsten Flensburg

Thu, 07/21/2005 (All day)

The Cryptographic Services APIs are the topic of this issue of APIs by Example. These APIs let you encrypt and decrypt data on the iSeries without needing to purchase additional software or hardware.

The example code for this article uses the Rijndal algorithm that was selected as the Advanced Encryption Standard (AES) by the National Institute of Standards and Technology (NIST) with the approval of the Federal Information Process Standard (FIPS) 197. There's a link to FIPS-197 at the end of this article.

The Cryptographic Services APIs became available by installing PTFs very soon after the release of V5R2. Subsequently, IBM also added the new APIs to the API documentation for that release. I will provide a link to the documentation at the end of this article as well.

Here are PTFs to look for to ensure that your V5R2 machine has the Cryptographic Services APIs on board:

```
V5R2 PTFs: SI10060 - Common Cryptographic APIs
           SI10105 - Common Cryptographic API includes
           MF31101 - Common Cryptographic API fix
```

For later releases, the Cryptographic Services APIs are part of the base install.

The cryptographic algorithms supported by the APIs in question are also dependent on the presence of the IBM iSeries software product 5722-AC3 -- Cryptographic Access Provider 128-bit for AS/400. You can use the CL command Display Software Resources (DSPSFWRSC) to verify that this product is installed. If not, it can be ordered free of charge from IBM or your business partner. Please note, however, that outside of the U.S. this product can be subject to U.S. export regulations.

For releases earlier than V5R2, the \_CIPHER MI built-in offers a subset of the cryptographic algorithms made available by the Cryptographic Services APIs. I've added a link below to the V5R1 documentation for that MI built-in, just in case.

AES replaces the old Data Encryption Standard (DES) algorithm as the encryption standard algorithm and is therefore an obvious choice for this example. To offer a simple interface to the APIs, I have packaged them in the CBX139 service program and predefined some of the API parameter settings. You can of course alter these settings any way you find appropriate for your own use.

To give you an idea of how the encryption and decryption process works, I have also written a small test program that runs three test scenarios:

1. The encryption and decryption process is run based on a predefined data string and key string.

2. The encryption and decryption process is run based on a data string specified by you and an API generated key string.
3. The encryption and decryption process is run based on the data string from test 2 and a 16-byte key string specified by you in the format of 32 hex nibbles. Nibbles are a character representation of the 4-bit sequences making up half a byte, as in the following 16 byte string example:

5FAC48BB687443BC06E611977A0D8C76

Please note that AES is a block cipher, and the encrypted string size is therefore a multiple of the specified block size, which for AES can be either 16, 24, or 32 bytes. In this example, I have used a block size of 16 bytes. For input data string sizes between 1 and 16 bytes, an output cipher string of 16 bytes is returned; input data strings between 17 and 32 bytes return a 32-byte output string; and so on, with the next size input data string always returning an output string that is up to the next 16 byte boundary.

If you want to store an encrypted value in a data file, you should define the length of the field to hold the cipher string in accordance with the above implementation. Let's say you want to encrypt and store a 19-digit credit card number. In this case, you would need a 32-byte field to hold the encrypted string. Likewise, a 35-letter name field would require a 48-byte field to hold the cipher string being returned from the encryption process.

Each example outputs information about the various steps it takes in a window on your screen. An example of this window follows:

```

.....
:
:   Data string . . : Very secret text string
:   Key string   . . : 5FAC48BB687443BC06E611977A0D8C76
:   Cipher string . : iôîMà iëU "Û: .Öè è $7 ÛÏÜ©¥
:   Key string   . . : 5FAC48BB687443BC06E611977A0D8C76
:   Clear text   . . : Very secret text string
:
:                                     Bottom
:   F12=Cancel
:
:.....

```

The first line displays the text string to be encrypted and the second line the encryption key string in an external, hexadecimal format. As a result of the encryption process, you see the encrypted string in the third line.

Next the process is reversed, using the decryption key displayed in the fourth line, which of course must be the same as the one used to encrypt the text string. The result, and verification, of the encryption and decryption test is shown in the fifth line.

Starting debug against the CBX139T program and stepping through the test examples using command key F10 is another recommended way to study the work of the Cryptographic Services APIs and the other functions included in the test program. To jump into the execution of the subprocedures, press F22 when reaching the subprocedure statements.

If you consider using cryptography in your applications, please note that cipher key administration and cipher key secrecy is the fundamental basis of doing so successfully. If a cipher key is compromised in any way, the information that it protects will no longer be secured against unauthorized access.

V5R3 brought a broad variety of enhancements to the cryptographic services APIs, including a whole set of APIs for key generation as well as a number of APIs dealing with the so called Cryptographic Context concept.

A cryptographic context is a symbolic representation (token) of either a cryptographic algorithm and state or a cryptographic key. Once created, the context token can be used as input to various cryptographic APIs, but only within the same job in which it was created. Since only the token is passed on to the cryptographic processes, the risk of exposing cryptographic algorithm and key values is potentially reduced.

In a future issue of APIs by Example, I will demonstrate some of the V5R3 Cryptographic Services API enhancements. In the mean time, please follow the link below to check out the V5R3 documentation.

This APIs by Example includes the following sources:

CBX139 -- Cryptographic services service program

CBX139B -- Service program binder source

CBX139T -- Test cryptographic services service program

Compilation instructions are found in the source headers -- please note that from this example and on I will be using binder language to define service program exports.

V5R1 \_CIPHER MI built-in documentation:

[http://publib.boulder.ibm.com/iserics/v5r1/ic2924/tstudio/tech\\_ref/mi/CIPHER.htm](http://publib.boulder.ibm.com/iserics/v5r1/ic2924/tstudio/tech_ref/mi/CIPHER.htm)

V5R2 Cryptographic Services APIs documentation:

<http://as400bks.rochester.ibm.com/iserics/v5r2/ic2924/info/apis/catcrypt.htm>

V5R3 Cryptographic Services APIs documentation:

<http://publib.boulder.ibm.com/infocenter/iserics/v5r3/ic2924/info/apis/catcrypt.htm>

FIPS-197 documentation:

<http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf>

This article demonstrates the following APIs:

Encrypt data (Qc3EncryptData) API:

<http://as400bks.rochester.ibm.com/iserics/v5r2/ic2924/info/apis/qc3encdt.htm>

Decrypt data (Qc3DecryptData) API:

<http://as400bks.rochester.ibm.com/iserics/v5r2/ic2924/info/apis/qc3decdt.htm>

Generate Pseudorandom Numbers (Qc3GenPRNs) API:

<http://as400bks.rochester.ibm.com/iserics/v5r2/ic2924/info/apis/qc3genprns.htm>

Display Long Text (QUILNGTX) API:

<http://as400bks.rochester.ibm.com/iserics/v5r2/ic2924/info/apis/quilngtx.htm>

Send Program Message (QMHSNDPM) API:

<http://publib.boulder.ibm.com/iserics/v5r2/ic2924/info/apis/QMHSNDPM.htm>

Receive Program Message (QMHRCVPM) API:

<http://as400bks.rochester.ibm.com/iserics/v5r2/ic2924/info/apis/QMHRCVPM.HTM>

You can retrieve the source code for this API example from

[http://www.pentontech.com/IBMContent/Documents/article/51236\\_28\\_CryptoServices.zip](http://www.pentontech.com/IBMContent/Documents/article/51236_28_CryptoServices.zip).

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