Article
Jon Sue-Ho · Jan 17, 2019 4m read

Format of public key when using RSAEncrypt method of %SYSTEM.Encryption or $System.Encryption.RSAEncrypt()

Howdy, Developer Community!

Here ' s a fun little formatting problem you may run into when trying to use the RSAEncrypt method of %SYSTEM.Encryption (also useable as $System.Encryption.RSAEncrypt()!), which is documented here:

https://docs.intersystems.com/latest/csp/documatic/%25CSP.Documatic.cls?...

This uses either a certificate or public key per the documentation. Quote:

Certificate/PublicKey - Either

An X.509 certificate containing the RSA public key to be used for encryption, in PEM encoded or binary DER format.

or

The RSA public key to be used for encryption, in PEM encoded format.

Note that the length of the plaintext can not be greater than the length of the modulus of the RSA public key contained in the certificate minus 42 bytes.

End quote.

When trying to use this with a public key rather than a certificate, it may not work. Specifically, you might get an empty string instead of an encrypted string as your output, and if you check $System.Encryption.RSAGetLastError(), you may see something like this:

EncryptionError: error:0D0680A8:asn1 encoding routines:ASN1CHECKTLEN:wrong

tag;error:0D06C03A:asn1 encoding routines:ASN1D2IEXPRIMITIVE:nested asn1

error;error:0D08303A:asn1 encoding routines:ASN1TEMPLATENOEXPD2I:nested asn1

error;error:0906700D:PEM routines:PEMASN1readbio:ASN1 lib;

What ' s going on?

Here ' s another near part of the documentation for this method right at the top that indirectly reveals some neat things. Quote:

This method performs RSA encryption as specified in PKCS #1 v2.1: RSA Cryptography Specifications, section 7 Encryption Schemes.

End quote.

Turns out, your public key might be in pem format, but turns out there's all sorts of encodings that have a pem format version. This method requires pkcs#1 pem format, which we'll show you how to recognize in the example below.

If you generated a public RSA key with openSSL from an existing private key, you probably did it with this command:

```
openssl rsa -in "/Pathto/PrivateKeyFile" -pubout
```

Actually, more likely you wrote the output straight to a file like this.

```
openssl rsa -in "/Pathto/PrivateKeyFile" -pubout > "/Pathto/PublicKeyFile"
```

Guess what! That output will have pkcs#8 pem format, which looks like this:

```
-----BEGIN PUBLIC KEY-----

SomeLongStringofAsciiCharacters…

-----END PUBLIC KEY-----
```

What we actually want is pkcs#1 pem format, which looks like this:

```
-----BEGIN RSA PRIVATE KEY-----

SomeVerySimilarLookingbutActuallyDifferentLongStringofAsciiCharacters…

-----END RSA PRIVATE KEY-----
```

This link I found illustrates the differences in ascii armor between pkcs#1 public keys, pkcs#8, and x509 public keys:

https://blog.ndpar.com/2017/04/17/p1-p8/

You might try to convert between this just by editing the ascii armor manually, but that ' s not the right way to do it since it will leave the actual contents in pkcs#8 format.

Instead, here ' s the openssl command to convert public key from pkcs#8 to pkcs#1 format:

```
openssl rsa -pubin -in "/Pathto/PublicKeyFileinpkcs8 " -RSAPublicKey_out > "/Pathto/P
ublicKeyFileinpkcs1"
```

(The internet documents this here: https://stackoverflow.com/questions/18039401/how-can-i-transform-between...)

Now you should be good to go to use that public key for RSA Encryption.

To recap, given an rsa private key, you can generate a pkcs#8 public key using this command:

```
openssl rsa -in "/Pathto/PrivateKey.txt" -pubout > "/Pathto/PublicKeyinpkcs8.txt"
```

Then convert the pkcs#8 public key to pkcs#1 with this command:

```
openssl rsa -pubin -in "/Pathto/PublicKeyFileinpkcs8 " -RSAPublicKey_out > "/Pathto/P
ublicKeyFileinpkcs1"
```

For convenience, here ' s some example code for reading a public key file into a stream then using it to encrypt a string:

```
set pubKeyFileName = "/Pathto/PublicKeyFileinpkcs1"

Set objCharFile = ##class(%FileCharacterStream).%New()

Set objCharFile.Filename = pubKeyFileName

Set pubKey = objCharFile.Read()

Set encryptedString = $System.Encryption.RSAEncrypt("StringtoBeEncrypted", pubKey)

write "encryptedString: ", encryptedString
```

Unencrypting is similar:

```
set privKeyFileName = "/Pathto/PrivateKeyFile"

Set privobjCharFile = ##class(%FileCharacterStream).%New()

Set privobjCharFile.Filename = privKeyFileName

Set privKey = privobjCharFile.Read()

set unencryptedString=$System.Encryption.RSADecrypt(encryptedString,privKey)

write "unencryptedString: ", unencryptedString
```

#Encryption #Caché

Source URL:https://community.intersystems.com/post/format-public-key-when-using-rsaencrypt-method-systemencryption-or-systemencryptionrsaencrypt