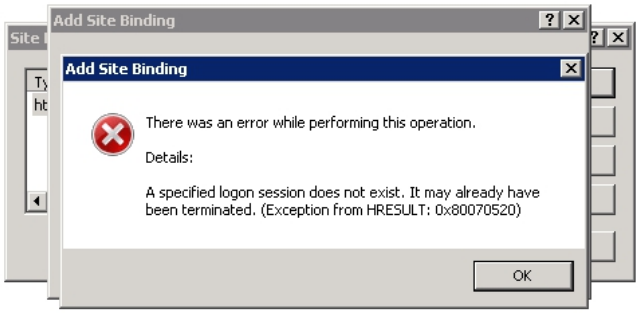
Binding SSL in IIS causes "A specified logon session does not exist. It may already have been terminated" error

There's maybe a zillion computer issues out there which are so damn weird that most people scratch their head in misbelief.
One of these weirdnesses just crossed my way a few days ago.

It all started with a very simple task: One of my engineers had to renew an SSL certificate for an IIS-hosted website.
After receiving the renewed SSL certificate and importing it into the certificate store, he tried to bind the SSL port on the IIS website with the new certificate.

What a bummer when he discovered, that this would always result in the following error, no matter what and how often he tried:



All investigation about this issue were dead ends.
Sure, there is a lot of information about this issue available on the net.

some suggest exporting and reimporting the certificate]
An msdn article suggests, that the certificate does not support digital signature use (hmmm ... well ... an officially signed certificate intended for HTTPS use!?!? especially one, where Certificate Manager SAID that is was intended for that particular purpose!?)]
Another one cares about the private key settings while importing the key/cert pair]

To make this short: A lot of posts, a lot of people having the same issue, a lot of suggestions, neither of which worked for us :-(

There's even the official KB981506 at Microsoft which tries to fix this for IIS 7 through a hotfix.

As with other possible solutions out there, there's people for whom this worked, while it didn't for others. Neither did it for us.

To make this even more worse: Using the very same private key/certificate pair on another server worked perfectly!

So, with all that information I thought about the export/import way being well worth another try - using a slight variation.
This takes also into account that the MSDN article mentioned above states the possibility of a wrong "key usage" token.

Therefore I made sure that I got a proper export from Certificate Manager into PKCS#12 (.pfx or .p12 extensions) including both the private key and the certificate for both the renewed and the previous period.

Afterwards I copied the files over to a UNIX machine and converted them from PKCS#12 file into PEM format using this openssl command:

```
openssl pkcs12 -in previous_cert_key.pfx -out previous_cert_key.pem -nodes
openssl pkcs12 -in renewed_cert_key.pfx -out renewed_cert_key.pem -nodes
```

Then I took a look at them X.509 headers of the PEM files. They looked as follows for the previous key/cert pair:

```
Bag Attributes
    Microsoft Local Key set:
    localKeyID: 01 00 00 00
    friendlyName: {29F1929F-FC74-412B-ACF1-45BEEC51631A}
    Microsoft CSP Name: Microsoft RSA SChannel Cryptographic Provider
Key Attributes
    X509v3 Key Usage: 10
```

And this is what it looked like for the renewed key/cert pair:

```
Bag Attributes
    Microsoft Local Key set:
    localKeyID: 01 00 00 00
    friendlyName: le-220133ff-668b-4e6a-946e-2c6581baeb86
    Microsoft CSP Name: Microsoft Enhanced Cryptographic Provider v1.0
Key Attributes
    X509v3 Key Usage: 80
```

Besides some rather to be ignored differences, the most notable and even as important one are the "X509v3 Key Usage" tokens, as they differ for the both key/cert pairs.

While the key usage 10 (should read 0x10) denotes an encipherment-elligible certificate, a key usage of 80 for the renewed certificate denotes it as valid for signing purposes only.

This however makes not much sense, as it should be encipherment-elligible, otherwise our TLS/SSL won't work at all.

Maybe this is the root cause of the certificate, which could not be properly used for TLS/SSL port binding on IIS?

Let's have a deeper look at the X.509 extended attributes, which can be reviewed like this:

```
openssl x509 -in previous_cert_key.pem -inform PEM -text
openssl x509 -in renewed_cert_key.pem -inform PEM -text
```

The extended key usage attributes read identically for both key/cert pairs:

X509v3 Key Usage: critical
Digital Signature, Key Encipherment
X509v3 Extended Key Usage:
TLS Web Server Authentication, TLS Web Client Authentication

So we obviously had a mismatch between the X.509 default and the respective extended attributes.

I got the feeling, that the MSDN article was indeed right about the "mismatch" reason. Especially if digging around the X.509 RFC, which stated some special considerations about when and how particular usage key tokens apply.

Basically to say is that conflicting values between the default and the extended key usage attributes may render a certificate useless in some cases, as they the flags are ignored in some cases.

So in this case I suppose that due to a somehow wrong default key usage attribute the certificate was still valid, but it was not elligible for envipherment use as the attributes mismatched each other.

To prove my assumption I then converted the PEM file of the renewed certificate back into a PKCS#12 file using this command, as already stated in a previous post:

openssl pkcs12 -export -in renewed_cert_key.pem -inkey renewed_cert_key.pem -out reconverted_cert_key.pfx

Obviously this caused the default key usage attribute to be fixed up in some way.
After importing the reconverted PKCS#12 file into Cert Manager, rebinding port 443 with the new certificate worked properly and SSL was usable from that point on.

To conclude from my experience: The root cause for this was a most-likely malformed default key usage attribute, which didn't match up to the extended key usage attributes.
As to my understanding, it may or may not be a software bug in IIS 7 on Windows Server 2008, which messed up binding the certificate to the port because it couldn't handle this situation properly. It is however unknown, why the hotfix didn't work and it must be assumed, that my case does not really apply to the KB issue.
As seen from the import on a Windows Server 2008 R2 host with IIS 7.5, the renewed certificate DID work properly there, even without tampering around with openssl conversion in the first. This might be due to a proper implementation which demands that the extended attributes are preferred over the default attributes.
Additionally, the successful export+openssl-conversion+reimport seems to support this assumption, as it did work afterwards on the Windows 2008 / IIS 7 host.
Here we must also consider that openssl did remove the default key usage attribute upon reconversion to PKCS#12 format.

After all, here's yet another possible way to work around this issue. This worked for me. Your mileage may vary, and most likely will ;-)