OpenSSL for Windows Developer's Guide

Trizen Systems, Inc.
120 International Parkway
Suite 220
Heathrow FL 32746

http://www.trizen.com
http://www.visualSSL.com

Zaremba@trizen.com

Trizen's OpenSSL Documentation and Developer's Guide Disclaimer

Contents

Introducing OpenSSL

"The OpenSSL Project is a collaborative effort to develop a robust, commercial-grade, full-featured, and Open Source toolkit implementing the Secure Sockets Layer (SSL v2/v3) and Transport Layer Security (TLS v1) protocols as well as a full-strength general purpose cryptography library. The project is managed by a worldwide community of volunteers that use the Internet to communicate, plan, and develop the OpenSSL toolkit and its related documentation."

-OpenSSL Organization

Before you Begin

This guide makes several assumptions: 1) You have a Visual C++ compiler version 6.0 or higher. 2) You are an experienced programmer using C++. 3) You have familiarity with the MS-DOS operating system and command line utilities.

The manual is intended for developers interested in using the OpenSSL toolkit and interested in using the OpenSSL libraries within their applications. It is also useful for gaining insight into Public Key Infrastructure (PKI) and Certificate Authority management as well as the usefulness of the OpenSSL command line tools.

What's Included in the Developer's Guide?

The OpenSSL for Windows Developer's Guide is made up of the following chapters:

Building the OpenSSL Libraries	This chapter focuses on Downloading and Building OpenSSL
Creating a Certificate Authority	Describes Creating, Issuing, and Signing Certificates within Windows
Using s_server & s_client	Describes using the two most important utilities in OpenSSL
OpenSSL Utilities	Using other OpenSSL Utilities
Authenticode	Digitally Sign your applications, use an SSL Server Certificate

Building the OpenSSL Libraries

In order to use the OpenSSL toolkit you must download it from the OpenSSL web site. This is fairly straightforward.

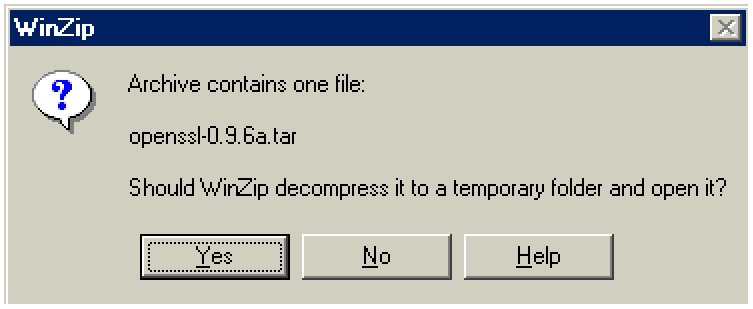
1) Visit http://www.openssl.org/source and download the latest (non-engine) tarball. A tarball, UNIX related, is similar to a zip file. Save the file anywhere on your hard drive, but remember its location.

2) Download and/or purchase WinZip at http://www.winzip.com. We recommend you spend the $40 to purchase a licensed copy of WinZip.

3) Go to the location where you downloaded and saved the openssl zip/tar file. If you are using WinZip as your registered zip utility, it should have a WinZip icon. Just double click on the icon.

openssl-0.9.6a
.tar.gz

3.1) If you see the following dialog, just press Yes to Continue:

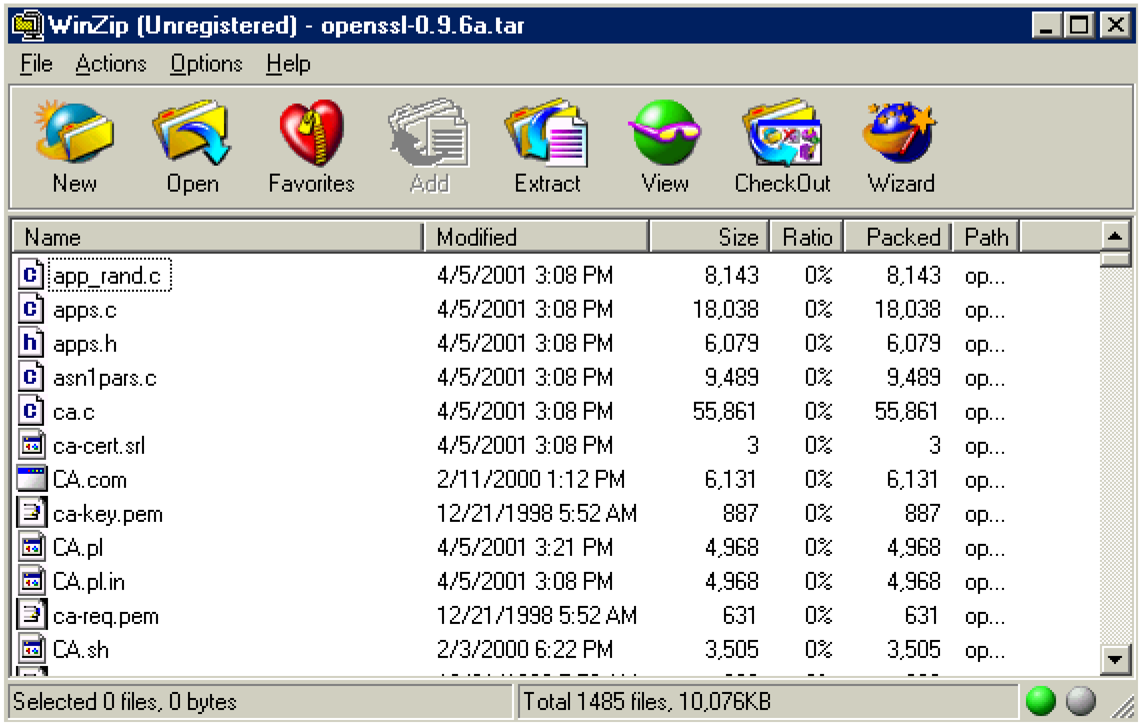


Archive contains one file:

openssl-0.9.6a.tar

Should WinZip decompress it to a temporary folder and open it?

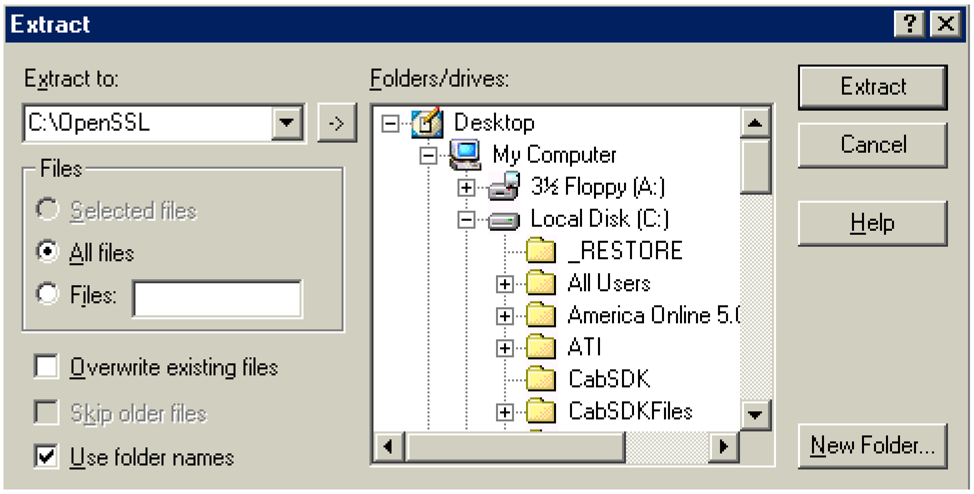
3.2) You should see the following dialog. Notice that this is a very large file. Also note that there are almost 1500 files associated with the tarball.


WinZip (Unregistered) - openssl-0.9.6a.tar
File Actions Options Help
New
Open
Favorites
Add
Extract
View
CheckOut
Wizard

Name	Modified	Size	Ratio	Packed	Path
app_rand.c	4/5/2001 3:08 PM	8,143	0%	8,143	op...
apps.c	4/5/2001 3:08 PM	18,038	0%	18,038	op...
apps.h	4/5/2001 3:08 PM	6,079	0%	6,079	op...
asn1pars.c	4/5/2001 3:08 PM	9,489	0%	9,489	op...
ca.c	4/5/2001 3:08 PM	55,861	0%	55,861	op...
ca-cert.srl	4/5/2001 3:08 PM	3	0%	3	op...
CA.com	2/11/2000 1:12 PM	6,131	0%	6,131	op...
ca-key.pem	12/21/1998 5:52 AM	887	0%	887	op...
CA.pl	4/5/2001 3:21 PM	4,968	0%	4,968	op...
CA.pl.in	4/5/2001 3:08 PM	4,968	0%	4,968	op...
ca-req.pem	12/21/1998 5:52 AM	631	0%	631	op...
CA.sh	2/3/2000 6:22 PM	3,505	0%	3,505	op...

Selected 0 files, 0 bytes   Total 1485 files, 10,076KB

4) Extract the files to C:\OpenSSL by selecting Actions | Extract. Create the OpenSSL directory in the "Extract to:" field by making your screen look similar to the next image then click Extract.



Extract to: C:\OpenSSL

Files: Selected files / [✓ All files] / Files:

Overwrite existing files: ☐

Skip older files: ☐

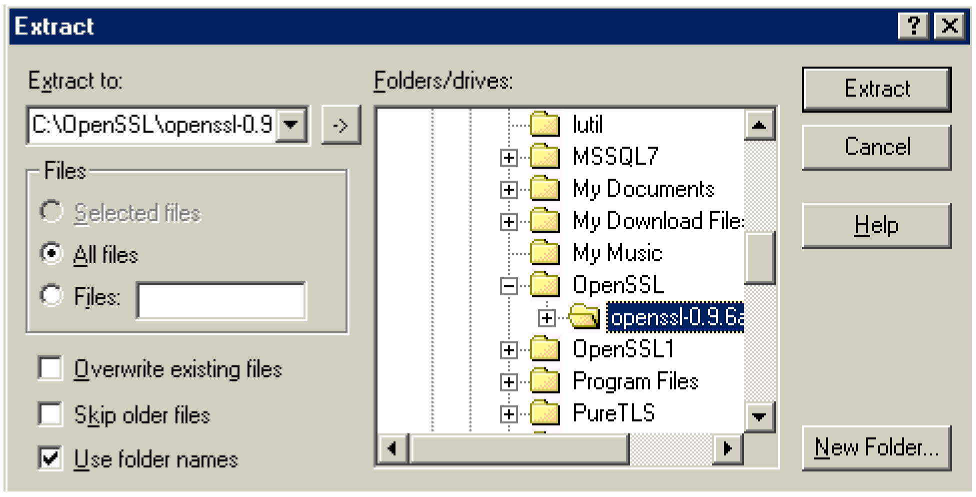
Use folder names: ☑

4.1) This new OpenSSL directory will now house a new folder titled:

4.2) The latest version may have a **different version** number than the above, but it should be similar. You can close the WinZip application. But notice that there is no MSVC folder within the openssl-9.6.a folder. This will be installed later.

5) Download the MS Developer Studio Workspace by visiting http://www.iconsinc.com/~agray/ossldev/ and click on the NO-ASM Non Assembler version. This file should be VC6ossl096a.zip noting that version numbers may differ slightly. Assembler versions will have a 'masm' or 'nasm' included in the title—DO NOT DOWNLOAD these. Save this zip file anywhere on your hard drive by clicking on its link or Shift+click on its link, but remember its location. If the above website is unavailable then visit the OpenSSL website and look for related information, which should have a link.

6) Double click on the saved VC6ossl096a.zip file (the version number may be different) which will open up the WinZip utility. You must extract this file into the above **openssl-0.9.6a** directory created earlier in step 4.



Extract to: C:\OpenSSL\openssl-0.9

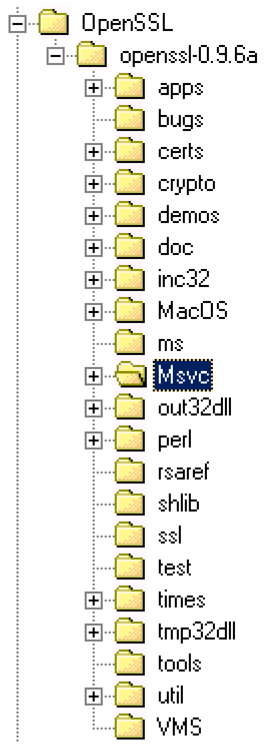
Files
( ) Selected files
[✓ All files]
( ) Files:

[ ] Overwrite existing files
[ ] Skip older files
[✓] Use folder names

Folders/drives:

```
lutil
MSSQL7
My Documents
My Download File
My Music
OpenSSL
    openssl-0.9.6a
OpenSSL1
Program Files
PureTLS
```

6.1) NOTE: Upon extracting the file into this directory you should have the following directory structure (with the new MSVC directory):

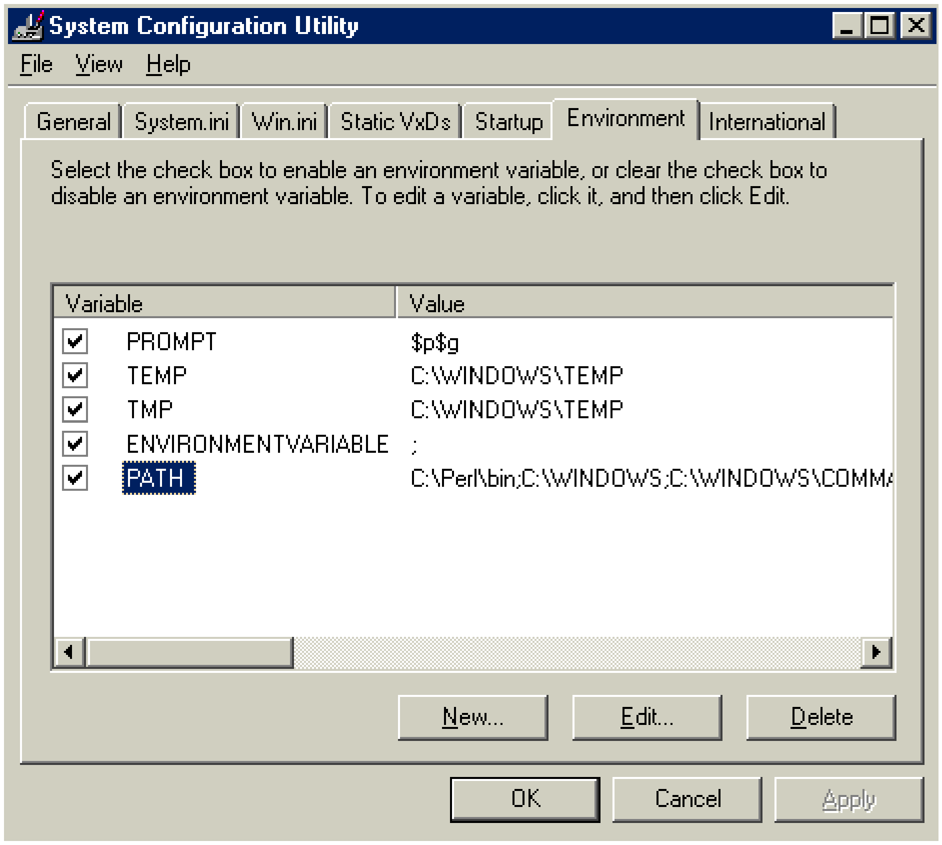


```
OpenSSL
    openssl-0.9.6a
        apps
        bugs
        certs
        crypto
        demos
        doc
        inc32
        MacOS
        ms
        Msvc
        out32dll
        perl
        rsaref
        shlib
        ssl
        test
        times
        tmp32dll
        tools
        util
        VMS
```

7) If you have not already done so, you must install Visual C++ 6.0. Remembering to run the VCVARS32.bat file. This file is located in the VC98 directory of the Microsoft Visual Studio folder. Just double click on the VCVARS32.bat file.

8) You must now download Perl for Windows. We have chosen ActivePerl and it can be downloaded at http://www.activestate.com/Products/ActivePerl/. Of course any Perl for Windows will work, but make sure that your PATH variable has the location of the Perl.exe file appended to the PATH variable string after you have installed

ActivePerl. This can be checked in Windows ME by going to Start | Run and typing MSCONFIG.EXE. You should see the following dialog, noting that the installation location for this example is *c:\Perl\bin*. This is the directory path to where the Perl.exe file was installed. By adding this location to your path variable you can issue a Perl command from any DOS prompt without having to be in the same directory as the Perl.exe file.


System Configuration Utility
File View Help
General System.ini Win.ini Static VxDs Startup Environment International

Select the check box to enable an environment variable, or clear the check box to disable an environment variable. To edit a variable, click it, and then click Edit.

	Variable	Value
✓	PROMPT	pg
✓	TEMP	C:\WINDOWS\TEMP
✓	TMP	C:\WINDOWS\TEMP
✓	ENVIRONMENTVARIABLE	;
✓	PATH	C:\Perl\bin;C:\WINDOWS;C:\WINDOWS\COMMA

New...
Edit...
Delete
OK
Cancel
Apply

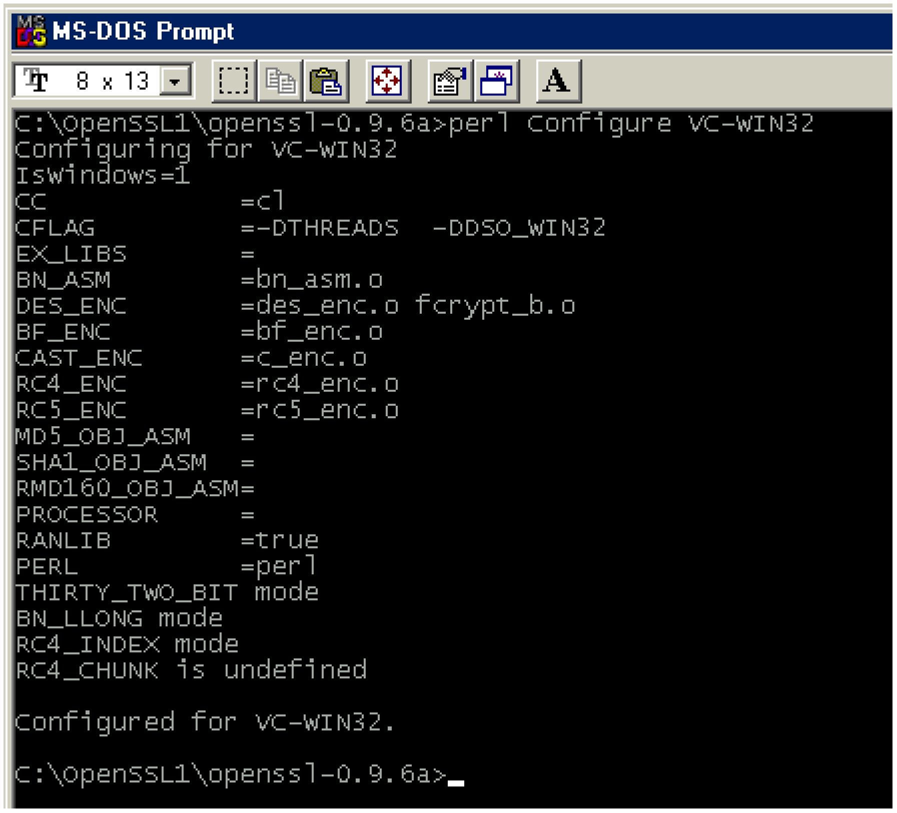
9) Notice the new C:\Perl\bin entry in the PATH variable. Again, this location represents this example. The location where you install Perl may differ slightly. Just make sure that the directory to the Perl.exe file that is installed with ActivePerl is appended somewhere in the PATH variable string. Windows NT users can access the PATH variable by right clicking on the *My Computer* icon, selecting the Advanced tab and then clicking on the Environment Variables button. For other operating systems, please see your Windows OS documentation for PATH variables.

10) Once you have a Perl program installed, you should copy the command.com (cmd.exe for NT) to the openssl-0.9.6a directory. This will allow you to run the necessary Perl scripts provided with the OpenSSL toolkit. The command.com or cmd.exe file can be found in your Windows or WINNT directory. Users of Windows NT/2000/XP must use cmd.exe.

11) Double click on the new command.com file (or cmd.exe) in your openssl-0.9.6a directory. At the command prompt type the following and press enter noting that this command is case sensitive:

C:\OpenSSL\openssl-0.9.6a>*Perl Configure VC-WIN32*

You should see something similar to the following screen:



```
C:\openSSL1\openssl-0.9.6a>perl Configure VC-WIN32
Configuring for VC-WIN32
IsWindows=1
CC              =cl
CFLAG           =-DTHREADS  -DDSO_WIN32
EX_LIBS         =
BN_ASM          =bn_asm.o
DES_ENC         =des_enc.o fcrypt_b.o
BF_ENC          =bf_enc.o
CAST_ENC        =c_enc.o
RC4_ENC         =rc4_enc.o
RC5_ENC         =rc5_enc.o
MD5_OBJ_ASM     =
SHA1_OBJ_ASM    =
RMD160_OBJ_ASM=
PROCESSOR       =
RANLIB          =true
PERL            =perl
THIRTY_TWO_BIT mode
BN_LLONG mode
RC4_INDEX mode
RC4_CHUNK is undefined

Configured for VC-WIN32.

C:\openSSL1\openssl-0.9.6a>_
```

11.1) If you don't see the above information after you type the *Perl Configure VC-WIN32* command you should review your steps and possibly delete the OpenSSL directory, extracting the files again and starting over. If you get an error that states:

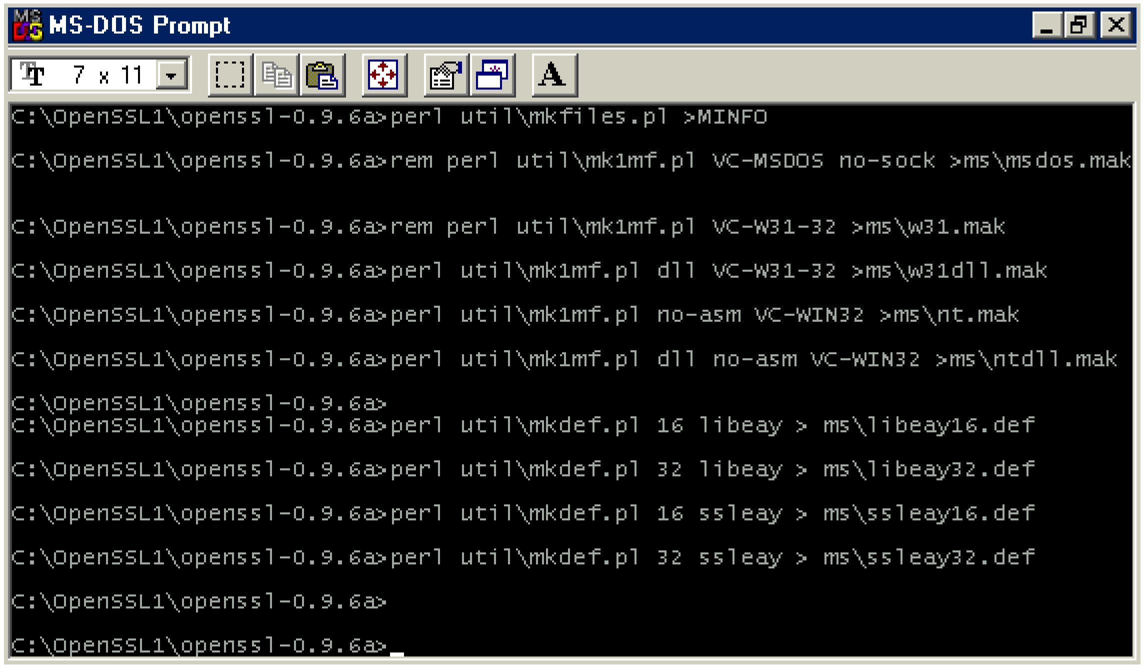
"'perl is not recognized as an internal or external operable program or batch file"

11.2) If you get the above message, then check to make sure you have installed Perl and that the directory location of Perl.exe is contained somewhere in the PATH system variable.

12) At the same command prompt type the following and press enter:

C:\OpenSSL\openssl1-0.9.6a>ms\do_ms.bat

You should see something similar to the following screen:



```
C:\OpenSSL1\openssl-0.9.6a>perl util\mkfiles.pl >MINFO

C:\OpenSSL1\openssl-0.9.6a>rem perl util\mk1mf.pl VC-MSDOS no-sock >ms\msdos.mak


C:\OpenSSL1\openssl-0.9.6a>rem perl util\mk1mf.pl VC-W31-32 >ms\w31.mak

C:\OpenSSL1\openssl-0.9.6a>perl util\mk1mf.pl dll VC-W31-32 >ms\w31dll.mak

C:\OpenSSL1\openssl-0.9.6a>perl util\mk1mf.pl no-asm VC-WIN32 >ms\nt.mak

C:\OpenSSL1\openssl-0.9.6a>perl util\mk1mf.pl dll no-asm VC-WIN32 >ms\ntdll.mak

C:\OpenSSL1\openssl-0.9.6a>
C:\OpenSSL1\openssl-0.9.6a>perl util\mkdef.pl 16 libeay > ms\libeay16.def

C:\OpenSSL1\openssl-0.9.6a>perl util\mkdef.pl 32 libeay > ms\libeay32.def

C:\OpenSSL1\openssl-0.9.6a>perl util\mkdef.pl 16 ssleay > ms\ssleay16.def

C:\OpenSSL1\openssl-0.9.6a>perl util\mkdef.pl 32 ssleay > ms\ssleay32.def

C:\OpenSSL1\openssl-0.9.6a>

C:\OpenSSL1\openssl-0.9.6a>_
```

12.1) If you don't see something similar to the above, you may have to start over making sure you have typed everything correctly and placed the extracted files in the right directories.

13) At the same command prompt type the following and press enter:

C:\OpenSSL\openssl-0.9.6a>*perl msvc\doinc.pl*

13.1) If all goes well, the screen should pause for a second and then basically return the command prompt. This command will create three (3) new directories in the openssl-0.9.6a directory:

- inc32
- out32dll
- tmp32dll

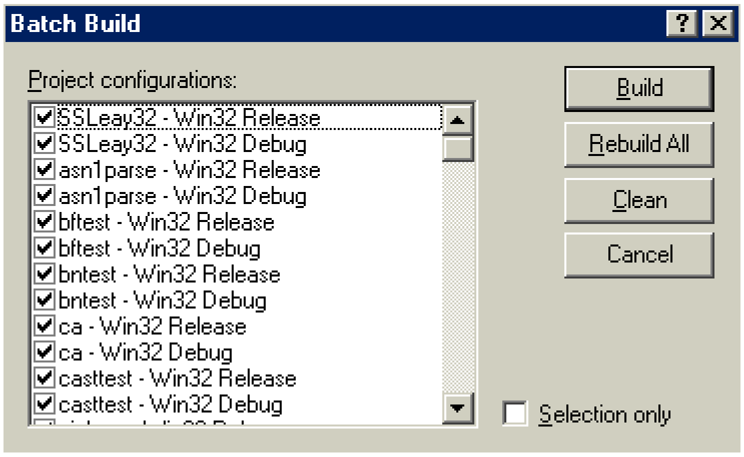
13.2) The most important of these is the out32dll directory which will have the Microsoft Visual C++ Release and Debug directories. Within these directories will be the OpenSSL libraries after a successful build in Visual C++. At this point you should check the openssl-0.9.6a directory to make sure these were successfully created.

13.3) If these directories do not appear in your openssl-0.9.6a then try starting from the beginning making sure you follow each step carefully.

14) Open Visual C++ and open the OpenSSL.dsw file in the newly created msvc directory within the openssl-0.9.6a directory. You may have to select *.dsw from the types of files to view in the Open project dialog.

15) You should see approximately 59 or so projects. The most important projects being the SSLeay32 and Libeay32 projects. Other important projects are the s_server and s_client projects.

16) At this point you should perform a batch build by selecting from the menu bar, Build | Batch Build. Press the *Rebuild All* button on the resulting form making sure that all projects are checked:



☑ SSLeay32 - Win32 Release
☑ SSLeay32 - Win32 Debug
☑ asn1parse - Win32 Release
☑ asn1parse - Win32 Debug
☑ bftest - Win32 Release
☑ bftest - Win32 Debug
☑ bntest - Win32 Release
☑ bntest - Win32 Debug
☑ ca - Win32 Release
☑ ca - Win32 Debug
☑ casttest - Win32 Release
☑ casttest - Win32 Debug

16.1) This will take upwards of 10 minutes to compile all the projects including the Debug and Release versions. You should hear several windows attention "beeps" as the compiler moves from project to project.

17) At this point the OpenSSL libraries: SSLeay32.dll and Libeay32.dll can be utilized in your applications. They can be found in the Release directory of the out32dll directory within the openssl-0.9.6a directory.

18) APPEND YOUR PATH VARIABLE TO POINT TO THE NEW RELEASE DIRECTORY UNDER THE OUT32DLL i.e. PATH=c:\OpenSSL\openssl-0.9.6a\out32dll\release. Make sure that the OpenSSL release directory for the correct version is contained in the PATH variable or you will not be able to execute the OpenSSL commands from anywhere but the release directory.

Creating Your Own Certificate Authority

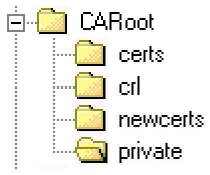
What is a Certificate Authority? A Certificate Authority is a clearinghouse of sorts that will provide a signed certificate that you may use in your applications. Some of the most notable are Verisign and Thawte which provide SSL certificates for use within web servers as well as Authenticode certificates for Microsoft ActiveX distribution. These certificates enable encryption as well as provide a "security blanket" of sorts that ensures a company is safe to communicate with.

By creating your own certificate authority, developers can utilize their own certificates in their applications. Visual 3270 and Visual SSL, products designed by Trizen all use certificates created by a Certificate Authority under a Trizen server. If you want to create your own certificates then you must create a CA structure on your network that can be accessed.

Of course you can get started right away by using the *server.pem* and/or *client.pem* files that come with OpenSSL. These files can be found in *c:\OpenSSL\openssl-0.9.6a\apps* directory.

Creating the Directory Structure

This is relatively painless. Create a directory named CARoot anywhere on your hard drive. For this documentation we are creating a new directory at c:\CARoot. Within this directory create four new directories so that your directory structure looks similar to the following:



Creating the Random File (.rnd)

First, copy the command.com or cmd.exe file to the *CARoot* directory. Next, create a .rnd file in the new private subdirectory—the tricky part is saving it without a prefix so that it is named ".rnd" rather than "random.rnd" or something similar. To do this in windows, double click on the command.com or cmd.exe file in the *CARoot* directory. At the MS-DOS (command) prompt type the following and press enter to create the ".rnd" file without a prefix:

```
C:\CARoot>edit private\.rnd
```

When the MS-DOS text edit fills the screen, just type in a bunch of data or open up a file of data such as an executable file, then save the file. You should now have a ".rnd" file located in the *CARoot\private* directory. If you don't have a ".rnd" file, then make sure you follow the directions exactly as outlined.

Creating the CA Key File

At this point you must make your Certificate Authority (CA) private key and certificate files. These files will be utilized when creating other certificates signed by your own Certificate Authority. You will be using the command prompt many times, so to save on navigation you should have already placed another copy of the command.com or cmd.exe file into:

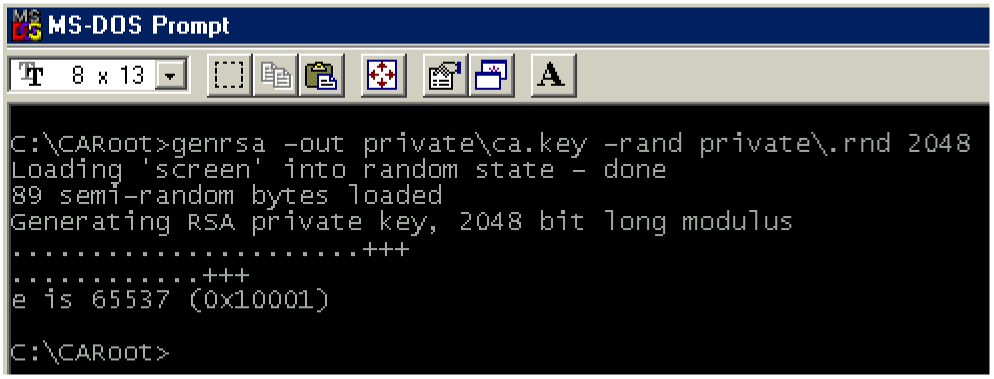
c:\CARoot

Double click on this file (command.com or cmd.com) unless its already open, and from the command prompt type the following and then press enter:

C:\CARoot>*genrsa –out private\ca.key –rand private\.rnd 2048*

NOTE: Switches and Options that are usually preceded by a '—' are used to specify certain attributes or properties of the command. There should be a space before the '—' and after the switch/option

The above command will generate an RSA Key file that will represent the CA Key file and will be used to generate public certificates. Double check to make sure the *ca.key* file is now in the *private* directory in the *CARoot* directory that was created earlier. If you get an error saying "bad command" then make sure you have modified your PATH environment variable to point to *c:\OpenSSL\openssl-0.9.6a\out32dll\release* directory (see Building the OpenSSL Libraries). If all has gone well then you should have seen something similar to the following screen shot:



```
C:\CARoot>genrsa -out private\ca.key -rand private\.rnd 2048
Loading 'screen' into random state - done
89 semi-random bytes loaded
Generating RSA private key, 2048 bit long modulus
.......................+++
............+++
e is 65537 (0x10001)

C:\CARoot>
```

If you want to add a password to your private key file, which is highly recommended, then perform the following command writing over the existing file:

C:\CARoot>*genrsa –out private\ca.key –rand private\.rnd –des3 2048*

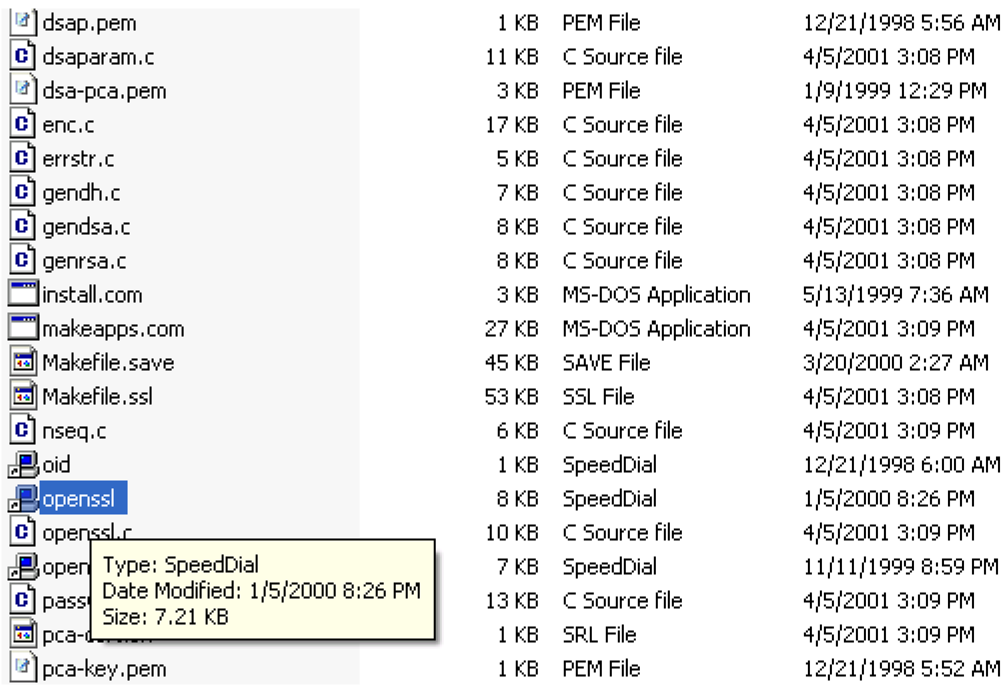
Notice that all we added was the *–des3* switch. If you would like to see more information about this command type:

```
C:\CARoot\genrsa –help
```

You may use this switch for any of the OpenSSL commands and comes in quite handy when you forget a switch option.

Creating the Config File (OpenSSL.cnf)

The Config file, named *OpenSSL.cnf,* is used for creating public certificates. The config file has all the information necessary to reproduce a public certificate based on the CA architecture and the CA Key file. A sample of a config file is provided in the OpenSSL apps directory. Unfortunately, Windows views any file with a .cnf extension as a *speed dial* file and may be tough to locate; however, look for a file named "openssl" without a visible extension with a "speed dial" association:



dsap.pem	1 KB	PEM File	12/21/1998 5:56 AM
dsaparam.c	11 KB	C Source file	4/5/2001 3:08 PM
dsa-pca.pem	3 KB	PEM File	1/9/1999 12:29 PM
enc.c	17 KB	C Source file	4/5/2001 3:08 PM
errstr.c	5 KB	C Source file	4/5/2001 3:08 PM
gendh.c	7 KB	C Source file	4/5/2001 3:08 PM
gendsa.c	8 KB	C Source file	4/5/2001 3:08 PM
genrsa.c	8 KB	C Source file	4/5/2001 3:08 PM
install.com	3 KB	MS-DOS Application	5/13/1999 7:36 AM
makeapps.com	27 KB	MS-DOS Application	4/5/2001 3:09 PM
Makefile.save	45 KB	SAVE File	3/20/2000 2:27 AM
Makefile.ssl	53 KB	SSL File	4/5/2001 3:08 PM
nseq.c	6 KB	C Source file	4/5/2001 3:09 PM
oid	1 KB	SpeedDial	12/21/1998 6:00 AM
openssl	8 KB	SpeedDial	1/5/2000 8:26 PM
openssl.c	10 KB	C Source file	4/5/2001 3:09 PM
open	7 KB	SpeedDial	11/11/1999 8:59 PM
pass	13 KB	C Source file	4/5/2001 3:09 PM
pca-	1 KB	SRL File	4/5/2001 3:09 PM
pca-key.pem	1 KB	PEM File	12/21/1998 5:52 AM

Type: SpeedDial
Date Modified: 1/5/2000 8:26 PM
Size: 7.21 KB

Copy this file from the OpenSSL apps directory to the CARoot directory. If not already open, double click on the command.com or cmd.exe file in the CARoot directory and type the following into the MS-DOS prompt:

```
C:\CARoot>edit openssl.cnf
```

This will bring up the MS-DOS editor with the openssl.cnf file opened for editing. The following snippet represents Trizen's configuration file:

```
# OpenSSL example configuration file.
# This is mostly being used for generation of certificate requests.
#

# This definition stops the following lines choking if HOME isn't
# defined.
HOME                = .
RANDFILE            = $ENV::HOME/.rnd

# Extra OBJECT IDENTIFIER info:
#oid_file           = $ENV::HOME/.oid
oid_section         = new_oids

# To use this configuration file with the "-extfile" option of the
# "openssl x509" utility, name here the section containing the
# X.509v3 extensions to use:
# extensions        =
# (Alternatively, use a configuration file that has only
# X.509v3 extensions in its main [= default] section.)

[ new_oids ]

# We can add new OIDs in here for use by 'ca' and 'req'.
# Add a simple OID like this:
# testoid1=1.2.3.4
# Or use config file substitution like this:
# testoid2=${testoid1}.5.6

####################################################################
[ ca ]
default_ca          = CA_default        # The default ca section

####################################################################
[ CA_default ]

d                    = .                      # Where everything is kept
certs               = $dir/certs            # Where the issued certs are kept
crl_di              = $dir/crl              # Where the issued crl are kept
database             = $dir/index.txt            # database index file.
new_certs_dir       = $dir/newcerts         # default place for new certs

c                                 a.c
serial               = $dir/serial               # The current serial number
crl                 = $dir/crl.pem          # The current CRL
priva      y         = $dir/private/c     y  # The private k
RANDFILE            = $dir/private/.rnd      # random data to use with certs
```

The most important components are the highlighted items and are discussed as follows:

1. dir: Should be "." which means "the current directory where this (openssl.cnf) configuration file is located."
2. certs: where issued or requested certificates should be saved.
3. crl_dir: Certificate Revocation Lists, those certificates that are should no longer be accepted.

4. database: a text (index.txt) file that will hold the increment of the certificates issued.
5. new_certs_dir: Signed certificates will be placed here (C:\CARoot\newcerts).
6. certificate: The CA Certificate derived from the CA Key (ca.crt).
7. serial: Each certificate is given a serial number as it is created, this keeps the increment.
8. private_key: The CA Key file. Keep this private (ca.key).
9. RANDFILE: The file that will be used in creating keys. It should be noted that some attacks and breaches of encryption have been accomplished by using the random number. Keep this private.

Other elements are used in developing public certificates, such as what questions will be asked, how they will be asked and other defaults. You may notice later in the configuration file that the default country may be AU, you should change this to reflect your country such as US. At this point make all the modification highlighted above to the openssl.cnf file in the MS-DOS editor.

After you have made the sample configuration file (openssl.cnf) match the above, save it in the c:\CARoot directory. You will now have an openssl.cnf file in the CARoot folder (directory).

Having problems opening the Configuration File

If you're having problems opening the file in Windows, don't be alarmed. From an MS-DOS prompt, navigate to the openssl.cnf file. At the prompt type the following and press enter:

```
C:\OpenSSL\openssl-0.9.6a\apps>edit openssl.cnf
```

This will bring up the file in an MS-DOS editor. You can now make your changes and then save the file to the CARoot directory.

Creating the Database (index.txt) and Serial files for the CA

You should take note of the database entry in the openssl.cnf Configuration file above which points to a file named *index.txt.* This file is made up of line entries for each new certificate signed by the CA. To create the index file, type the following at the C:\CARoot file then press enter:

```
C:\CARoot>touch .\index.txt
```

NOTE: Touch is a program that will set the file's timestamp to the current system time. If you don't have the touch.exe file or equivalent, don't worry, just create a text file named *index.txt* in the *CARoot* directory.

You must now create the serial file. This file can be created by typing the following at the MS-DOS prompt and then press enter:

C:\CARoot>*echo 01 > serial*

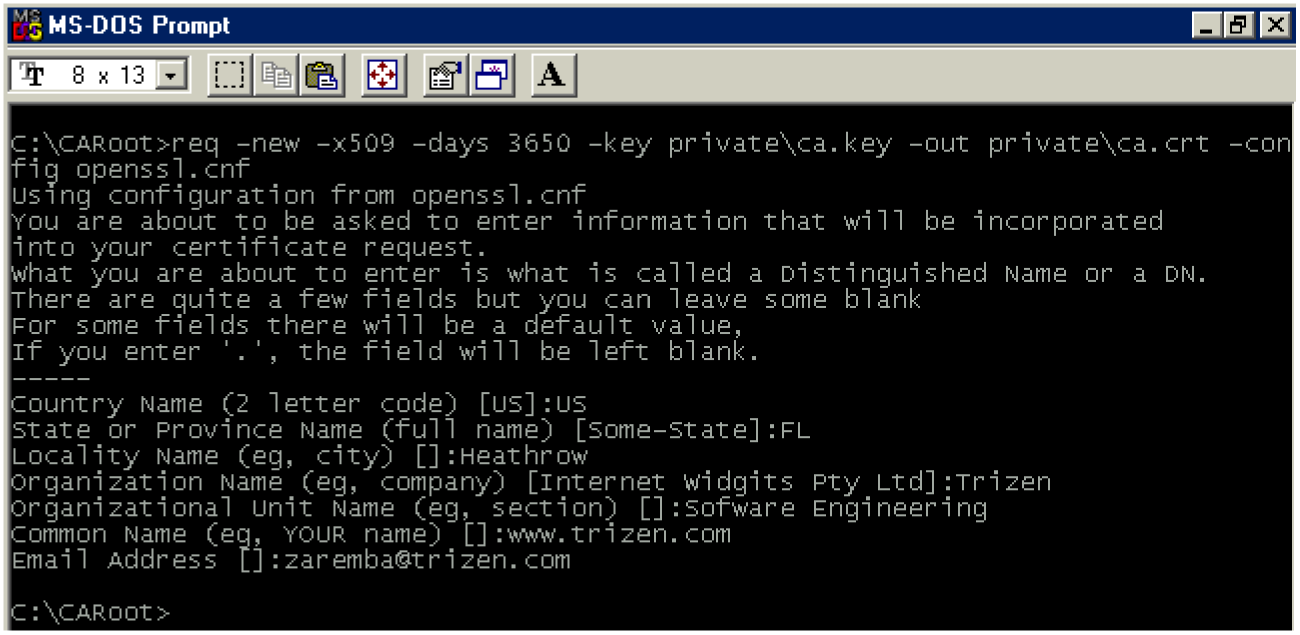
This will create a file named *serial* in the *CARoot* directory. You can open this file with WordPad or Notepad (right click on the file, select "Open with" and then select "WordPad"). If you open the file you will see it contains one line with the text "01" without the quotes. OpenSSL documentation uses double quotes around the '01'; however, this will not work in Windows. Ensure no quotes exist around the 01.

To summarize, you should now have two files in the *CARoot* directory named *serial* and *index.txt.* Note that these two files are documented in the Configuration file (openssl.cnf) with the names: database (for the index.txt) and serial (for the serial file). Make sure that these files are in the C:\CARoot directory and not any sub directories.

Creating the CA Certificate

At this point you can now successfully create the CA Certificate. Make sure you have updated the PATH environment variable to reflect the *[c]:\OpenSSL\openssl-0.9.6a\out32dll\release* files that have been created in the OpenSSL Visual C++ build (your path may differ slightly from the above). Double click on the command.com or cmd.exe file located in your *CARoot* directory. At the command prompt type the following and press enter:

C:\CARoot>*req –new –x509 –days 3650 –key private\ca.key –out private\ca.crt –config openssl.cnf*



```
C:\CARoot>req -new -x509 -days 3650 -key private\ca.key -out private\ca.crt -con
fig openssl.cnf
Using configuration from openssl.cnf
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [US]:US
State or Province Name (full name) [Some-State]:FL
Locality Name (eg, city) []:Heathrow
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Trizen
Organizational Unit Name (eg, section) []:Sofware Engineering
Common Name (eg, YOUR name) []:www.trizen.com
Email Address []:zaremba@trizen.com

C:\CARoot>
```

Notice in the above screen shot that we were asked a series of questions. As you create your certificates you should use your own CA information being aware that you should give your certificate a long expiration date such as the 10 year date above. Notice also the question text is derived from the configuration file (openssl.cnf). Also notice that nothing has been added to any other directory such as the *newcerts* directory after this command. Remember, when using the req.exe application above, you are creating a self signed certificate! A self signed certificate is "dummy" certificate and has not been signed by a Certificate Authority. If you have turned on any client or server verification, it will fail with a return code of (18), which means a self signed certificate. For CA Certificates, they are all self-signed because you are the top of the chain; however, when you create other certificates, you will sign those certificates with your CA information! The following represents what your <u>self signed</u> CA certificate should now look like if opened in WordPad:

```
-----BEGIN CERTIFICATE-----
MIIEhTCCA22gAwIBAgIBADANBgkqhkiG9w0BAQQFADCBjTELMAkGA1UEBhMCVVMx
CzAJBgNVBAgTAkZMMREwDwYDVQQHEwhIZWF0aHJvdzEPMA0GA1UEChMGVHJpemVu
MREwDwYDVQQLEwhTb2Z0d2FyZTEXMBUGA1UEAxMOd3d3LnRyaXplbi5jb20xITAf
BgkqhkiG9w0BCQEWEnphcmVtYmFAdHJpemVuLmNvbTAeFw0wMTEwMjAwMDAwNTZa
Fw0wMjEwMjAwMDAwNTZaMIGNMQswCQYDVQQGEwJVUzELMAkGA1UECBMCRkwxETAP
BgNVBAcTCEhlYXRocm93MQ8wDQYDVQQKEwZUcml6ZW4xETAPBgNVBAsTCFNvZnR3
YXJlMRcwFQYDVQQDEw53d3cudHJpemVuLmNvbTEhMB8GCSqGSIb3DQEJARYSemFy
ZW1iYUB0cml6ZW4uY29tMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA
yldagYx3PusRi1olOskdNX9YtnYAw+GL/uL+VyZb6qIqDQdiUnzZIHEQKCtMGcwN
Gc6IFU78aDfR8Hxf5D5mwe3uzlBh9zO4N9dZgKvW6B9eDuvY8EVuLuFq3JvZS5Wj
v9W3IwVf//2wauDyF/i/Vs+C8DcJLvV/iDLXgJVmAMJknum/nLBiz2rmNvdiANyB
gQJpiyAmESYAVcOpOhr5uuHDfxlmERpP14Ca3Vyhryp+se/T92sbhahupMBGufd4
fJ00O0JvuSuP4Wim6GETW4bUCvDIJuTvoN9Mq6Gq7IVmtEzJDCRY32xWguz/U26/
5UIzNGGeExwA6VWC8edDbQIDAQABo4HtMIHqMB0GA1UdDgQWBBQkmWA4HTAJeAO4
N3K9lIBpf8FUVTCBugYDVR0jBIGyMIGvgBQkmWA4HTAJeAO4N3K9lIBpf8FUVaGB
k6SBkDCBjTELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAkZMMREwDwYDVQQHEwhIZWF0
aHJvdzEPMA0GA1UEChMGVHJpemVuMREwDwYDVQQLEwhTb2Z0d2FyZTEXMBUGA1UE
AxMOd3d3LnRyaXplbi5jb20xITAfBgkqhkiG9w0BCQEWEnphcmVtYmFAdHJpemVu
LmNvbYIBADAMBgNVHRMEBTADAQH/MA0GCSqGSIb3DQEBBAUAA4IBAQCcn/7S9wQZ
/YytEEAnIXShIvwIRKeJF1Qj7IBN1nLRQclctZFs3tveLYPt2xr+AXcrgIq07JPN
ev33qpCxrBBDcpV5OSYHftZV7Ah3dbjR2Jb95nMWBIXONcZkeMbGjQjY7wMLnHvB
8E3vwTWdmVSFfFs7JT2BNhphuT5kuhqo6NEaHNtA9vKZGPBGJrgLF8PoXCk0J8vR
L5WP8z5Kqe1o4/9BT1WD0re6NtUh36XB1rNZfVuR01HshMNah5VL6UXRi5/pdttZ
uKdwIWpSs1YPIhR4MxFkHm4xif++pdN69ld6RHOpFs/1IS5rCEwBYrEPoSxjV7ud
QCAx3UW8vk6x
-----END CERTIFICATE-----
```

This is very standard stuff for self signed certificates, very simple with all the information encoded between the BEGIN CERTIFICATE and the END CERTIFICATE. At this point you have successfully created your CA Public Certificate and your CA Private Key files which should be kept in the *c:\CARoot\private directory*.

Creating a NEW Public Certificate and Key Pair

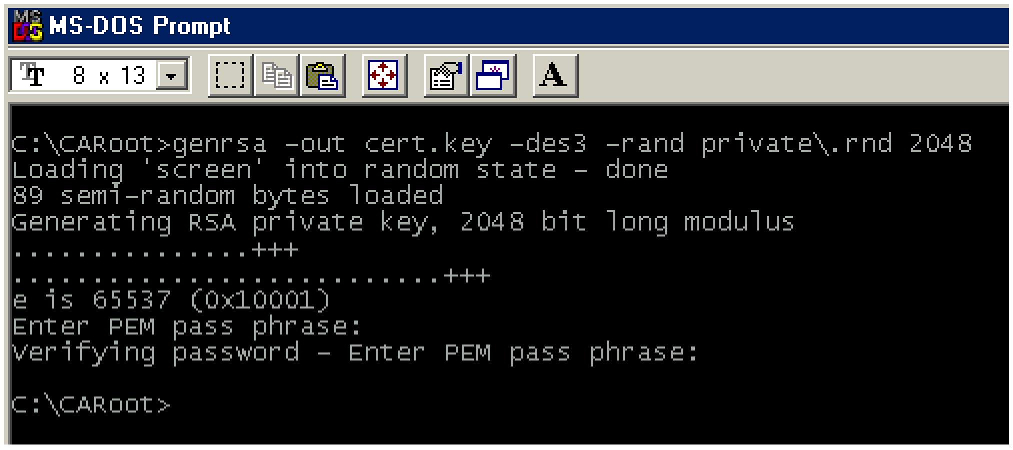
At this point you have a very basic CA structure but you can create certificates and sign them with your organization. We are going to walk you through creating a key pair:

1) Create a new Key file using the genrsa.exe command:

C:\CARoot>*genrsa –out cert.key –rand private\.rnd 2048*

2) Or Create a new Key file with a password using the genrsa command:

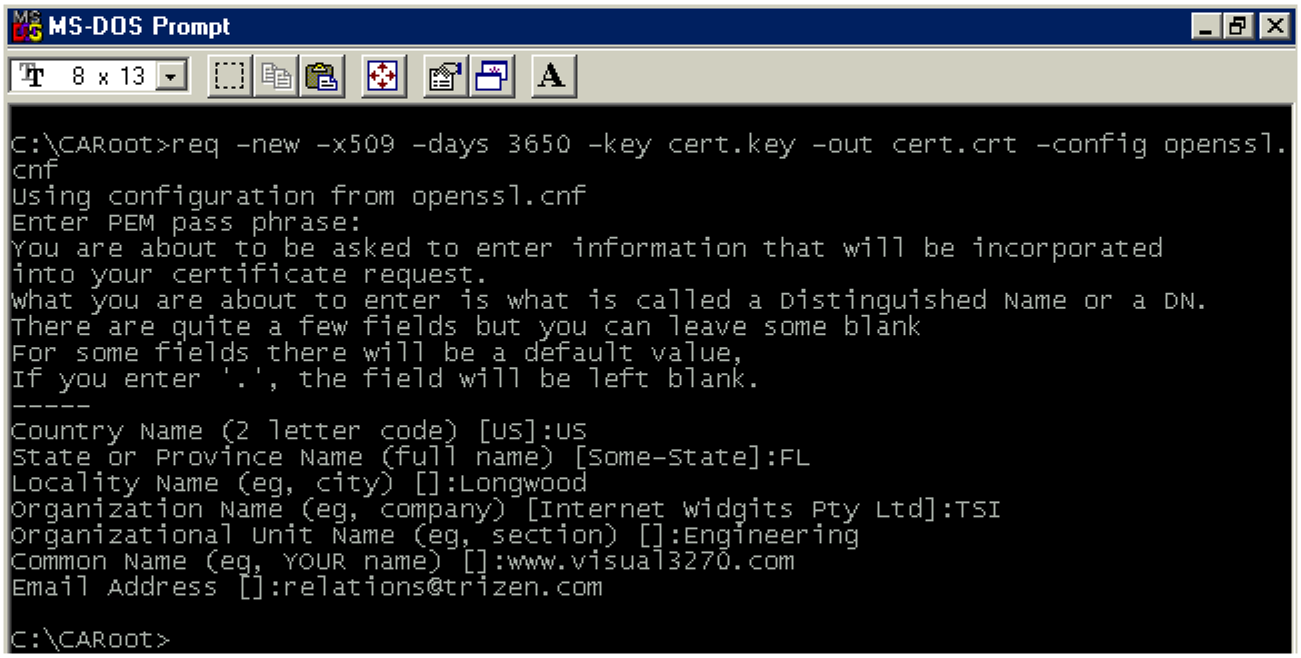
C:\CARoot>*genrsa –out cert.key –rand private\.rnd –des3 2048*



```
C:\CARoot>genrsa -out cert.key -des3 -rand private\.rnd 2048
Loading 'screen' into random state - done
89 semi-random bytes loaded
Generating RSA private key, 2048 bit long modulus
................+++
............................+++
e is 65537 (0x10001)
Enter PEM pass phrase:
Verifying password - Enter PEM pass phrase:

C:\CARoot>
```

If all goes well your should see something similar to the above screen shot. It is HIGHLY recommended that you use passwords to encrypt your key files for servers; however, there is no reason for client files to be encrypted with a password. If you forget your password you will have to recreate your certificate.

3) Create a Public Certificate using the req.exe command as detailed in the screen shot:



```
C:\CARoot>req -new -x509 -days 3650 -key cert.key -out cert.crt -config openssl.
cnf
Using configuration from openssl.cnf
Enter PEM pass phrase:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [US]:US
State or Province Name (full name) [Some-State]:FL
Locality Name (eg, city) []:Longwood
Organization Name (eg, company) [Internet Widgits Pty Ltd]:TSI
Organizational Unit Name (eg, section) []:Engineering
Common Name (eg, YOUR name) []:www.visual3270.com
Email Address []:relations@trizen.com

C:\CARoot>
```

If all goes well you should see the above screen shot. Note the questions that were asked and how they match with the openssl.cnf file modified earlier. You may at anytime open this configuration file and change the defaults. At this point you should have a cert.key and a cert.crt file in your CARoot directory. If these files do not exist, double check your work and try again. You should note that the new cert.crt file has not been signed and is very similar to the CA certificate at this point. The next step is to sign the certificate with your CA key file.

4) Sign the certificate using the following command:

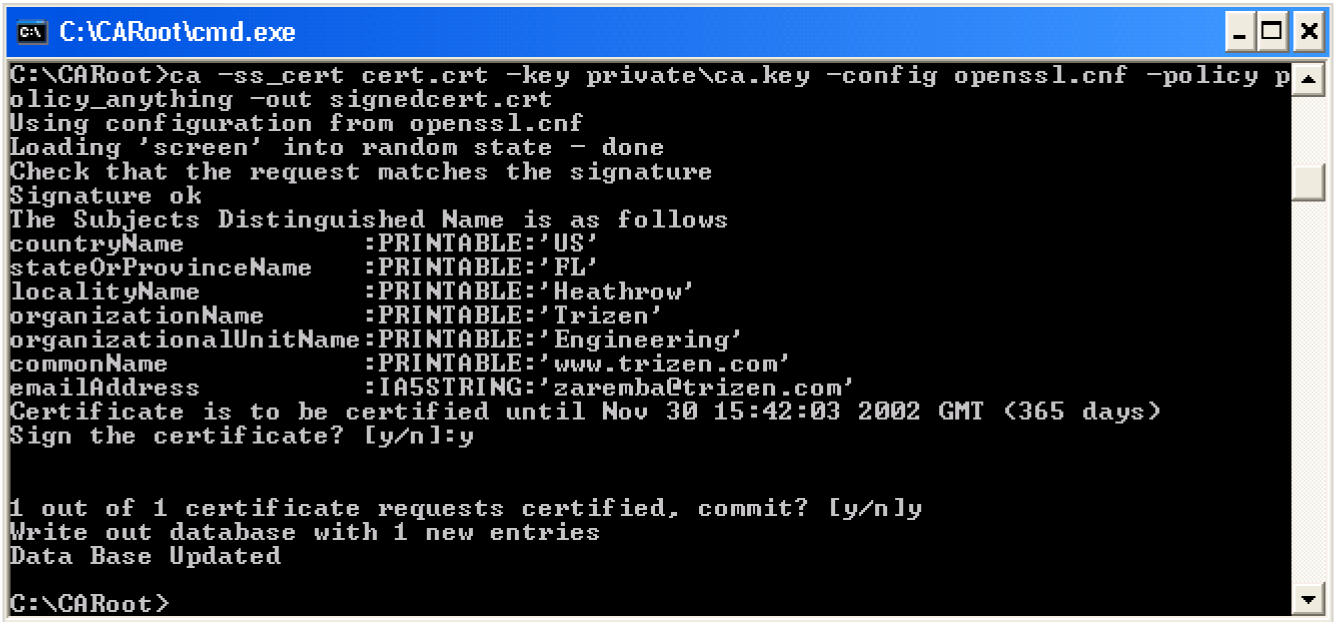
   C:\CARoot>*ca –ss_cert cert.crt –key private\ca.key –config openssl.cnf –policy policy_anything –out signedcert.crt*

Note: Do not use the *cert.key* as this is the newly generated Key (use it for generating a public key, not signing the key), you must use the CA Key file generated earlier to establish the certificate chain.

   In the above if we did not use the *–policy policy_anything* switch to accept the certificate as it currently exists the ca.exe command would complain. Once you have signed the certificate, one will be located in your *newcerts* directory as a record of the transaction with the serial number and a .pem extension i.e. *01.pem*. For each newly signed certificate, your newcerts directory will be updated, increasing the index number for each new certificate. The other location is in the CARoot directory and is the signedcert.crt file as specified in the *–out* parameter. Use this file, along with the cert.key file, in your OpenSSL applications.

Note: You may delete certificates the *index.txt* file if you want to recreate or resign the certificates. You may also change the sequence number in the *serial* file.

You should see a lot of information fly by when you issue the above command and then you should be asked if you want to sign the certificate? Just press Y and then press enter. You will then be asked if you want to “commit” the file. Just press Y and then press enter. You should see more information fly by and finally end with “database updated.”



```
C:\CARoot>ca -ss_cert cert.crt -key private\ca.key -config openssl.cnf -policy p
olicy_anything -out signedcert.crt
Using configuration from openssl.cnf
Loading 'screen' into random state - done
Check that the request matches the signature
Signature ok
The Subjects Distinguished Name is as follows
countryName           :PRINTABLE:'US'
stateOrProvinceName   :PRINTABLE:'FL'
localityName          :PRINTABLE:'Heathrow'
organizationName      :PRINTABLE:'Trizen'
organizationalUnitName:PRINTABLE:'Engineering'
commonName            :PRINTABLE:'www.trizen.com'
emailAddress          :IA5STRING:'zaremba@trizen.com'
Certificate is to be certified until Nov 30 15:42:03 2002 GMT (365 days)
Sign the certificate? [y/n]:y


1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated

C:\CARoot>
```

The above screen shot represents the typical "signing" process. If you did not see something similar to the above, then repeat the steps making sure you followed the examples explicitly.

If you check the *newcerts* directory you should see a new certificate in the directory with a *01.pem* filename (may be a different number, but it will be the highest). If you look at the file you should see something similar to the following:

```
Data:
        Version: 3 (0x2)
        Serial Number: 1 (0x1)
        Signature Algorithm: md5WithRSAEncryption
        Issuer: C=US, ST=FL, L=Heathrow, O=Trizen, OU=Software,
CN=www.trizen.com/Email=zaremba@trizen.com
        Validity
            Not Before: Oct 20 00:02:05 2001 GMT
            Not After : Oct 20 00:02:05 2002 GMT
        Subject: C=US, ST=FL, O=Trizen, OU=Software,
CN=www.trizen.com/Email=zaremba@trizen.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (2048 bit)
                Modulus (2048 bit):
                    00:ca:57:5a:81:8c:77:3e:eb:11:8b:5a:25:3a:c9:
                    1d:35:7f:58:b6:76:00:c3:e1:8b:fe:e2:fe:57:26:
                    5b:ea:a2:2a:0d:07:62:52:7c:d9:20:71:10:28:2b:
                    4c:19:cc:0d:19:ce:88:15:4e:fc:68:37:d1:f0:7c:
                    5f:e4:3e:66:c1:ed:ee:ce:50:61:f7:33:b8:37:d7:
                    59:80:ab:d6:e8:1f:5e:0e:eb:d8:f0:45:6e:2e:e1:
                    6a:dc:9b:d9:4b:95:a3:bf:d5:b7:23:05:5f:ff:fd:
                    b0:6a:e0:f2:17:f8:bf:56:cf:82:f0:37:09:2e:f5:
                    7f:88:32:d7:80:95:66:00:c2:64:9e:e9:bf:9c:b0:
                    62:cf:6a:e6:36:f7:62:00:dc:81:81:02:69:8b:20:
```

```
                    26:11:26:00:55:c3:a9:3a:1a:f9:ba:e1:c3:7f:19:
                    66:11:1a:4f:d7:80:9a:dd:5c:a1:af:2a:7e:b1:ef:
                    d3:f7:6b:1b:85:a8:6e:a4:c0:46:b9:f7:78:7c:9d:
                    34:3b:42:6f:b9:2b:8f:e1:68:a6:e8:61:13:5b:86:
                    d4:0a:f0:c8:26:e4:ef:a0:df:4c:ab:a1:aa:ec:85:
                    66:b4:4c:c9:0c:24:58:df:6c:56:82:ec:ff:53:6e:
                    bf:e5:42:33:34:61:9e:13:1c:00:e9:55:82:f1:e7:
                    43:6d
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            Netscape Comment:
                OpenSSL Generated Certificate
            X509v3 Subject Key Identifier:
                24:99:60:38:1D:30:09:78:03:B8:37:72:BD:94:80:69:7F:C1:54:55
            X509v3 Authority Key Identifier:

keyid:24:99:60:38:1D:30:09:78:03:B8:37:72:BD:94:80:69:7F:C1:54:55

DirName:/C=US/ST=FL/L=Heathrow/O=Trizen/OU=Software/CN=www.trizen.com/Email=zar
emba@trizen.com
                serial:00

    Signature Algorithm: md5WithRSAEncryption
        7c:18:87:74:ad:a1:c3:90:55:af:27:f4:87:0c:4a:bb:5a:0d:
        f1:87:a4:2d:dc:a4:c1:16:c3:02:21:9b:4c:56:61:fe:ce:a1:
        4a:cb:fc:de:92:57:c6:87:63:38:ba:de:56:d6:3a:62:4c:c5:
        cb:41:75:44:61:f5:f7:d9:fa:9f:ab:35:9e:c3:90:96:b1:0b:
        1a:47:b3:4d:2c:15:60:e0:95:ed:98:b4:31:8a:7e:77:f3:41:
        0e:05:91:f4:1e:1d:fd:74:e6:7c:61:b7:16:f6:ec:a9:b7:d1:
        aa:b0:23:e1:42:8a:c5:50:b3:c8:a9:f3:69:d5:49:1e:95:d3:
        21:12:ba:6a:e9:35:f2:c5:74:15:d9:a7:d5:71:47:9a:a3:fe:
        5b:11:14:50:fe:d9:eb:76:f2:21:1e:88:ac:7c:4e:ea:19:f8:
        74:d8:0f:0b:97:af:3c:ce:95:79:b5:6a:b0:67:c0:0c:33:3a:
        fd:32:9d:a1:4c:0b:6e:21:b5:de:c4:e6:2f:6d:07:70:7d:31:
        0c:3f:7d:b6:4d:ac:d7:3e:94:4c:1c:26:a2:19:1e:ee:ff:da:
        73:73:49:38:c0:32:9e:22:8c:ed:d2:1f:3c:50:ba:89:df:d6:
        a5:ab:df:52:a2:79:7f:e0:91:d7:6a:45:44:9a:6a:d3:bc:a0:
        c0:18:ba:c0
-----BEGIN CERTIFICATE-----
MIIEnjCCA4agAwIBAgIBATANBgkqhkiG9w0BAQQFADCBjTELMAkGA1UEBhMCVVMx
CzAJBgNVBAgTAkZMMREwDwYDVQQHEwhIZWF0aHJvdzEPMA0GA1UEChMGVHJpemVu
MREwDwYDVQQLEwhTb2Z0d2FyZTEXMBUGA1UEAxMOd3d3LnRyaXplbi5jb20xITAf
BgkqhkiG9w0BCQEWEnphcmVtYmFAdHJpemVuLmNvbTAeFw0wMTEwMjAwMDAyMDVa
Fw0wMjEwMjAwMDAyMDVaMHoxCzAJBgNVBAYTAlVTMQswCQYDVQQIEwJGTDEPMA0G
A1UEChMGVHJpemVuMREwDwYDVQQLEwhTb2Z0d2FyZTEXMBUGA1UEAxMOd3d3LnRy
aXplbi5jb20xITAfBgkqhkiG9w0BCQEWEnphcmVtYmFAdHJpemVuLmNvbTCCASIw
DQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAMpXWoGMdz7rEYtaJTrJHTV/WLZ2
AMPhi/7i/lcmW+qiKg0HYlJ82SBxECgrTBnMDRnOiBVO/Gg30fB8X+Q+ZsHt7s5Q
YfczuDfXWYCr1ugfXg7r2PBFbi7hatyb2UuVo7/VtyMFX//9sGrg8hf4v1bPgvA3
CS71f4gy14CVZgDCZJ7pv5ywYs9q5jb3YgDcgYECaYsgJhEmAFXDqToa+brhw38Z
ZhEaT9eAmt1coa8qfrHv0/drG4WobqTARrn3eHydNDtCb7krj+FopuhhE1uG1Arw
yCbk76DfTKuhquyFZrRMyQwkWN9sVoLs/1Nuv+VCMzRhnhMcAOlVgvHnQ20CAwEA
AaOCARkwggEVMAkGA1UdEwQCMAAwLAYJYIZIAYb4QgENBB8WHU9wZW5TU0wgR2Vu
ZXJhdGVkIENlcnRpZmljYXRlMB0GA1UdDgQWBBQkmWA4HTAJeAO4N3K9lIBpf8FU
VTCBugYDVR0jBIGyMIGvgBQkmWA4HTAJeAO4N3K9lIBpf8FUVaGBk6SBkDCBjTEL
MAkGA1UEBhMCVVMxCzAJBgNVBAgTAkZMMREwDwYDVQQHEwhIZWF0aHJvdzEPMA0G
A1UEChMGVHJpemVuMREwDwYDVQQLEwhTb2Z0d2FyZTEXMBUGA1UEAxMOd3d3LnRy
aXplbi5jb20xITAfBgkqhkiG9w0BCQEWEnphcmVtYmFAdHJpemVuLmNvbYIBADAN
```

```
TFZh/s6hSsv83pJXxodjOLreVtY6YkzFy0F1RGH199n6n6s1nsOQlrELGkezTSwV
YOCV7Zi0MYp+d/NBDgWR9B4d/XTmfGG3FvbsqbfRqrAj4UKKxVCzyKnzadVJHpXT
BgkqhkiG9w0BAQQFAAOCAQEAfBiHdK2hw5BVryf0hwxKu1oN8YekLdykwRbDAiGb

IRK6auk18sV0Fdmn1XFHmqP+WxEUUP7Z63byIR6IrHxO6hn4dNgPC5evPM6VebVq


sGfADDM6/TKdoUwLbiG13sTmL20HcH0xDD99tk2s1z6UTBwmohke7v/ac3NJOMAy
niKM7dIfPFC6id/WpavfUqJ5f+CR12pFRJpq07ygwBi6wA==
-----END CERTIFICATE-----
```

If you look in the *index.txt* file in the *c:\CARoot* directory you should see the certificate information as a single line item. If you look in the *serial* file you should see "02" which represents the next certificate index (01 is the above certificate).

You should now distribute this newly created *signedcert.crt* file as the signed certificate. This certificate will work in the OpenSSL environment and will function properly with any application that uses the OpenSSL libraries; however, if you want to install this certificate in Windows then you must make some changes as documented in Adjusting the Certificates for Windows Operating Systems.

Signing Certificate Requests (Being a Certificate Authority)
In the previous example we demonstrated the signing of self signed "dummy" certificates. In this section we will show you how to take a certificate request and sign it with your credentials. Similar to Verisign, or Thawte, you create a certificate request usually from Internet Information Server and send it to them to "sign" and return the signed certificate for your use. Creating a certificate request from IIS is discussed in section **Creating PKCS#12 Certificates in Internet Information Server.** You may optionally issue the following command in the MS-DOS prompt to create a Certificate Signing Request:

```
C:\CARoot>req –newkey rsa:1024 –keyout private\dummy.key –out
        private\dummy.pem –config openssl.cnf
```

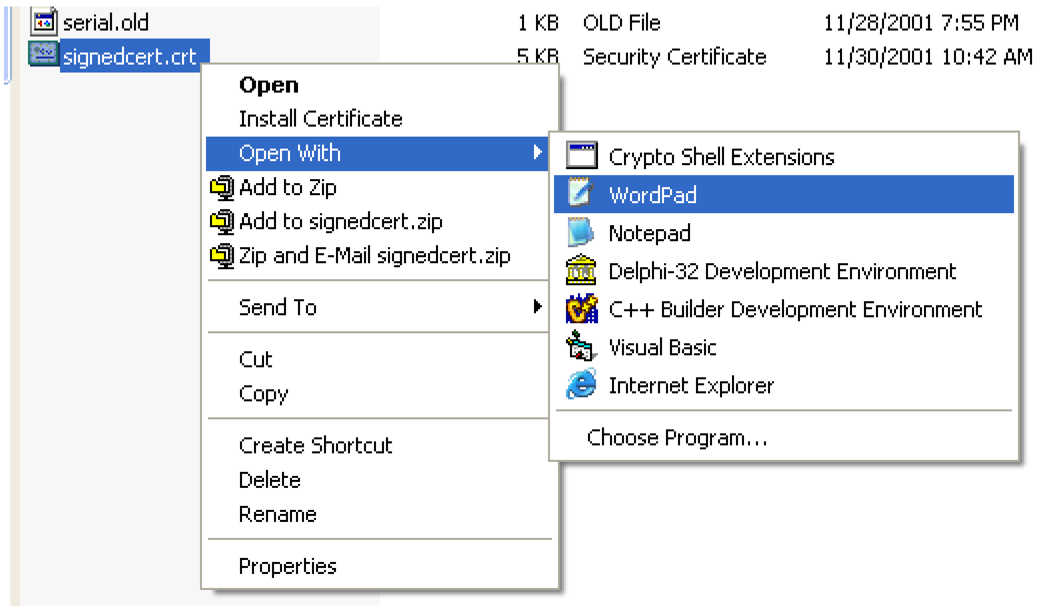
Note, that in order to create certificate requests, you have to have a Certificate Authority setup. This is sometimes very tough for those who don't want to be a CA. It is therefore recommended to create them from Internet Information Server. After getting the Certificate Signing Request file (or after creating it) you can issue the following command to sign the certificate:

```
C:\CARoot>ca –in certreq.txt –key private\ca.key –out newcerts\mycert.cer –policy
        policy_anything –config openssl.cnf
```

Note that the only difference between signing self-signed certificates and certificate request are the –ss_cert and –in switches respectively.

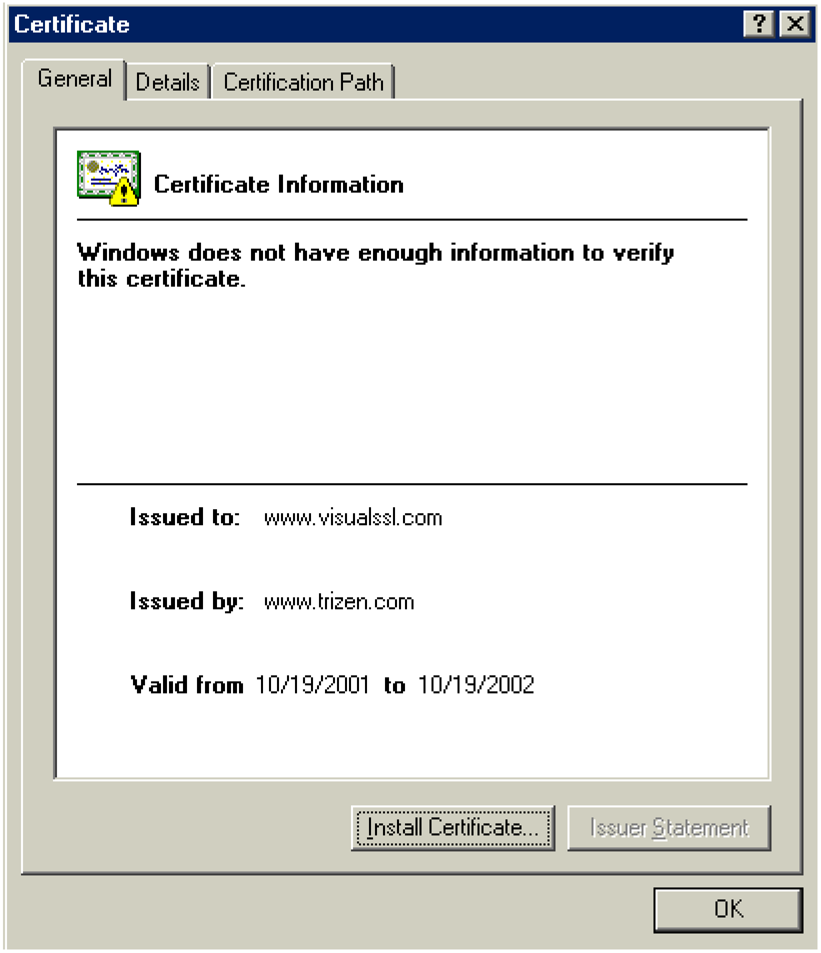
Adjusting the Certificates for Windows Operating Systems

Windows does not understand the .pem file as first created by OpenSSL. Windows does understand .crt files. You may have noticed that when you sign a certificate, its contents change, you don't just have the BEGIN CERTIFICATE and END CERTIFICATE you have all the information at the top as well. If you merely change the extension of the .pem file to .crt and then double click the file, Windows will give you an error. You must delete the entire contents of the file except for the data between and including the BEGIN CERTIFICATE and the END CERTIFICATE (similar to the self signed CA certificate above). Go ahead and right mouse click on the newly created signedcert.crt file and open the file with WordPad.



It is best to use WordPad when opening certificate files as NotePad will display all of the extra ASCII character codes and make editing the certificate difficult. Once the file has opened in WordPad, delete everything except for the data between and including the BEGIN CERTIFICATE and the END CERTIFICATE making sure that the BEGIN CERTIFICATE line is the first line and the END CERTIFICATE line is the last line. Save the file (don't worry if you mess up, you can recreate the certificate again, and optionally delete the entry from the index.txt file and serial file). Your file should now look very similar in nature to the ca.crt file.

When you do delete the header information, and then, if necessary, change the extension from .pem to .crt, you can then double click on the file you should see the following noting that your information may be slightly different:



Windows does not have enough information to verify this certificate.

Issued to: www.visualssl.com

Issued by: www.trizen.com

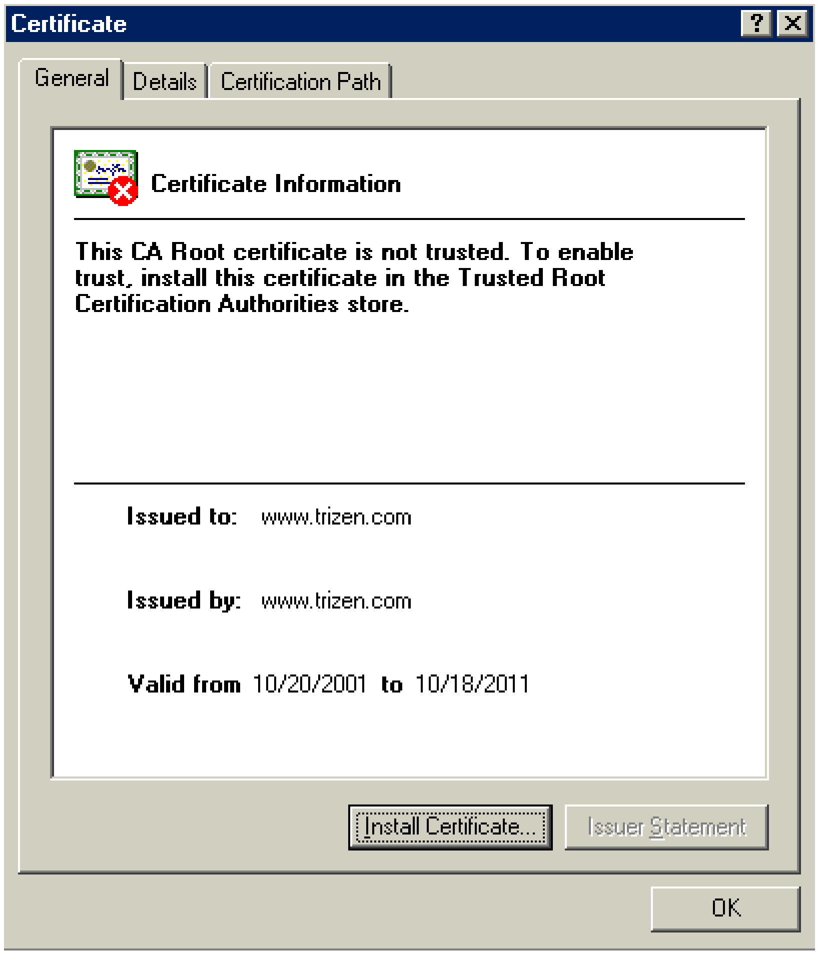
Valid from 10/19/2001 to 10/19/2002

You do have the option of installing the certificate into the Windows certificate database of certificates to trust, but is not necessary at this time. If you check the other tabs, you will notice all of the data you have entered for this signed certificate as well as information about the issuer which is the CA. You will also notice that Windows cannot find any information on this certificate. We will install the CA certificate in the next section.

Installing the CA Certificate into the Windows Operating System

For any signed certificate, you must take out the header information and have only the BEGIN CERTIFICATE and END CERTIFICATE information such as the information presented in the **Creating the CA Certificate** section. Open up the ca.crt file in the *c:\CARoot\private* directory by right mouse clicking on the file and then opening it up with WordPad. Make sure it looks like a standard certificate with no header information just ------BEGIN CERTIFICATE----- and ------END CERTIFICATE ----------.

If the CA certificate conforms with Windows .crt formatting, then you can simply double click on the certificate. When the dialog pops up, similar to the above screen shot, press the *Install Certificate…* button:



Certificate Information

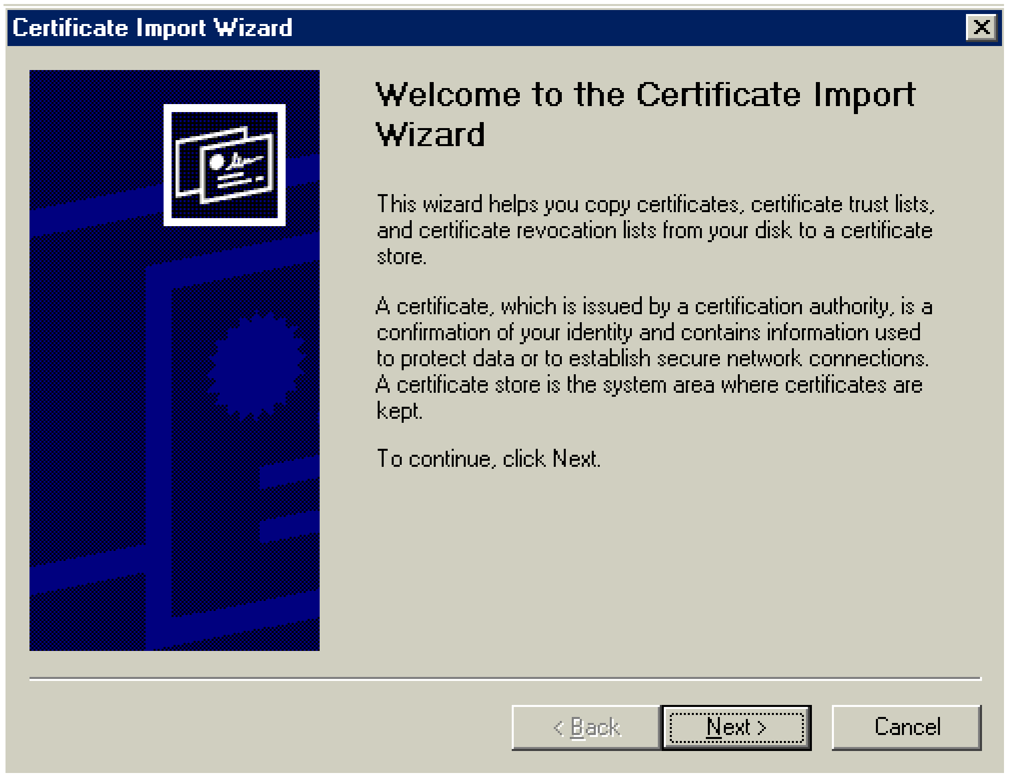
This CA Root certificate is not trusted. To enable trust, install this certificate in the Trusted Root Certification Authorities store.

Issued to: www.trizen.com

Issued by: www.trizen.com

Valid from 10/20/2001 to 10/18/2011

After pressing the *Install Certificate*… button you see the following dialog box:


Certificate Import Wizard

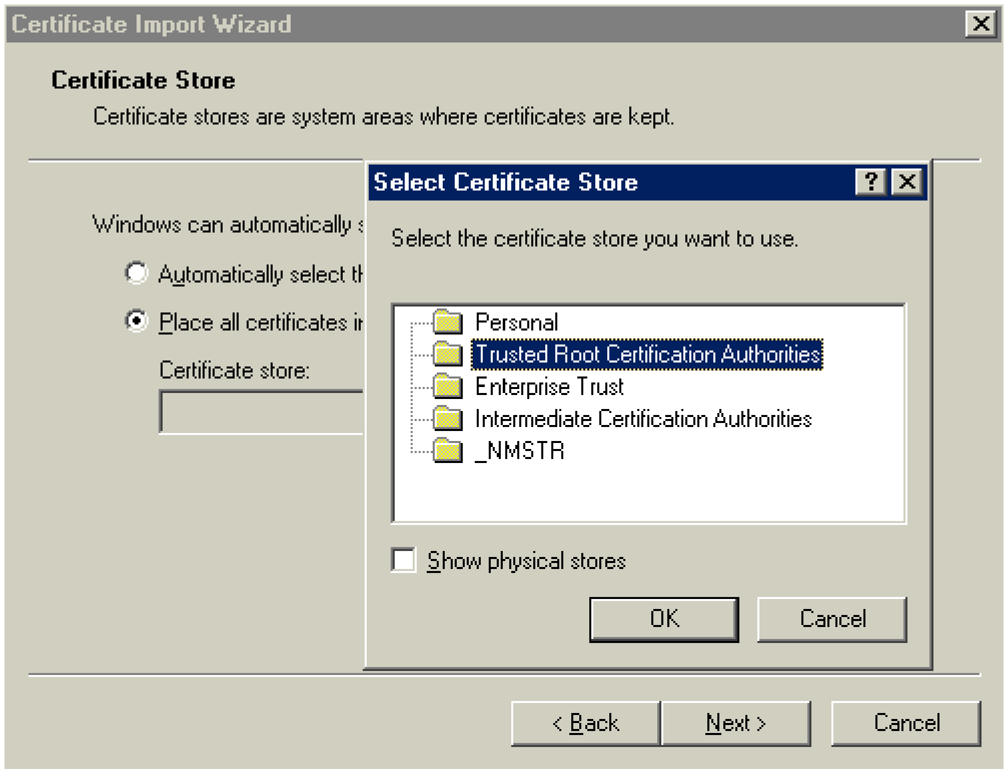
Welcome to the Certificate Import Wizard

This wizard helps you copy certificates, certificate trust lists, and certificate revocation lists from your disk to a certificate store.

A certificate, which is issued by a certification authority, is a confirmation of your identity and contains information used to protect data or to establish secure network connections. A certificate store is the system area where certificates are kept.

To continue, click Next.

Just Press *Next* > to continue


Certificate Import Wizard

Certificate Store

Certificate stores are system areas where certificates are kept.

Windows can automatically s

Automatically select th

Place all certificates ir

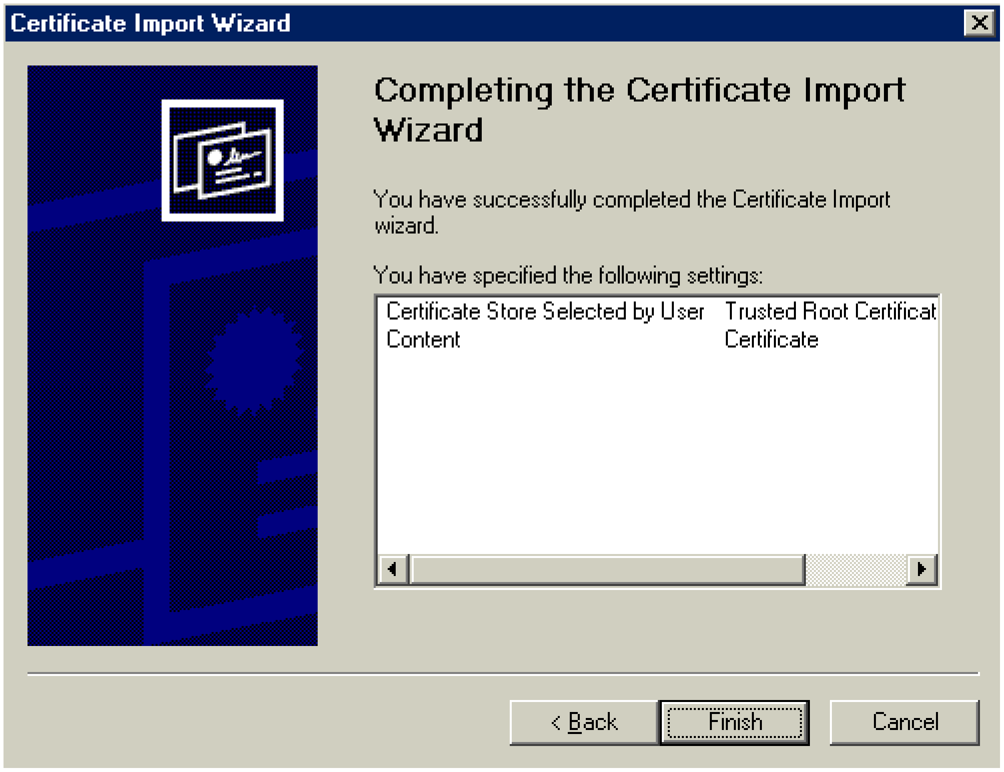
Certificate store:

Select Certificate Store

Select the certificate store you want to use.

Personal
Trusted Root Certification Authorities
Enterprise Trust
Intermediate Certification Authorities
_NMSTR
Show physical stores
OK
Cancel
< Back
Next >
Cancel

In the above dialog I have selected the *"Place all certificates in the following store"* and then pressed the *Browse…* button. When the Select Certificate Store popped up, I selected the *"Trusted Root Certificate Authorities"* store and pressed *Ok.* Press Next > to continue. You should see the following dialog:



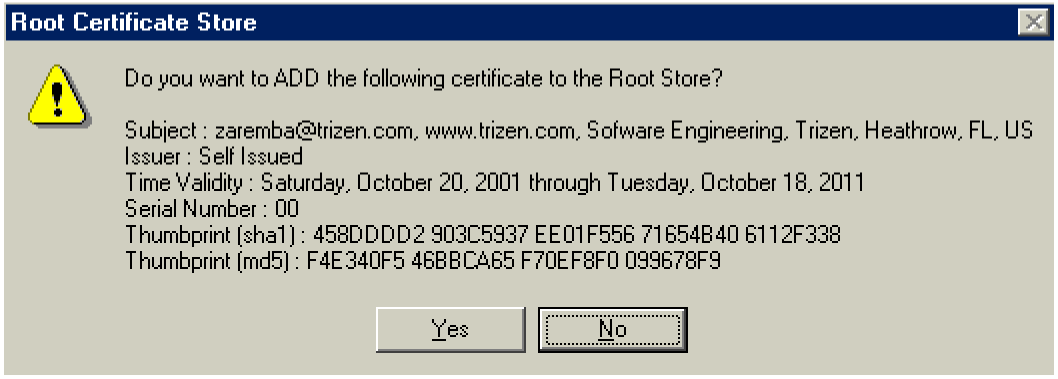
Completing the Certificate Import Wizard

You have successfully completed the Certificate Import wizard.

You have specified the following settings:

Certificate Store Selected by User	Trusted Root Certificat
Content	Certificate

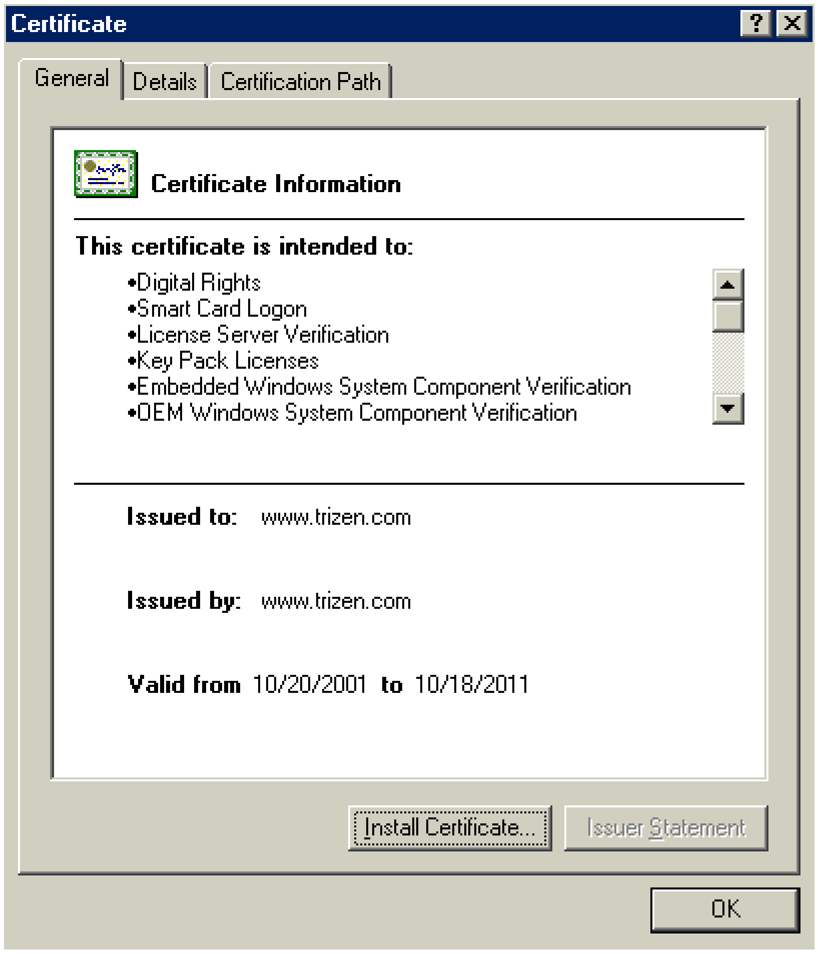
Just Press *Finish…* to complete the process:



Do you want to ADD the following certificate to the Root Store?

Subject : zaremba@trizen.com, www.trizen.com, Sofware Engineering, Trizen, Heathrow, FL, US
Issuer : Self Issued
Time Validity : Saturday, October 20, 2001 through Tuesday, October 18, 2011
Serial Number : 00
Thumbprint (sha1) : 458DDDD2 903C5937 EE01F556 71654B40 6112F338
Thumbprint (md5) : F4E340F5 46BBCA65 F70EF8F0 099678F9

Press *Yes* to continue and install the certificate. You have now installed the CA Root certificate as a trusted Certificate Authority. The next step will be to install any public certificates that have been signed by the CA Root certificate. You may notice something different the next time you double click on the CA certificate (ca.crt):



This certificate is intended to:

- Digital Rights
- Smart Card Logon
- License Server Verification
- Key Pack Licenses
- Embedded Windows System Component Verification
- OEM Windows System Component Verification

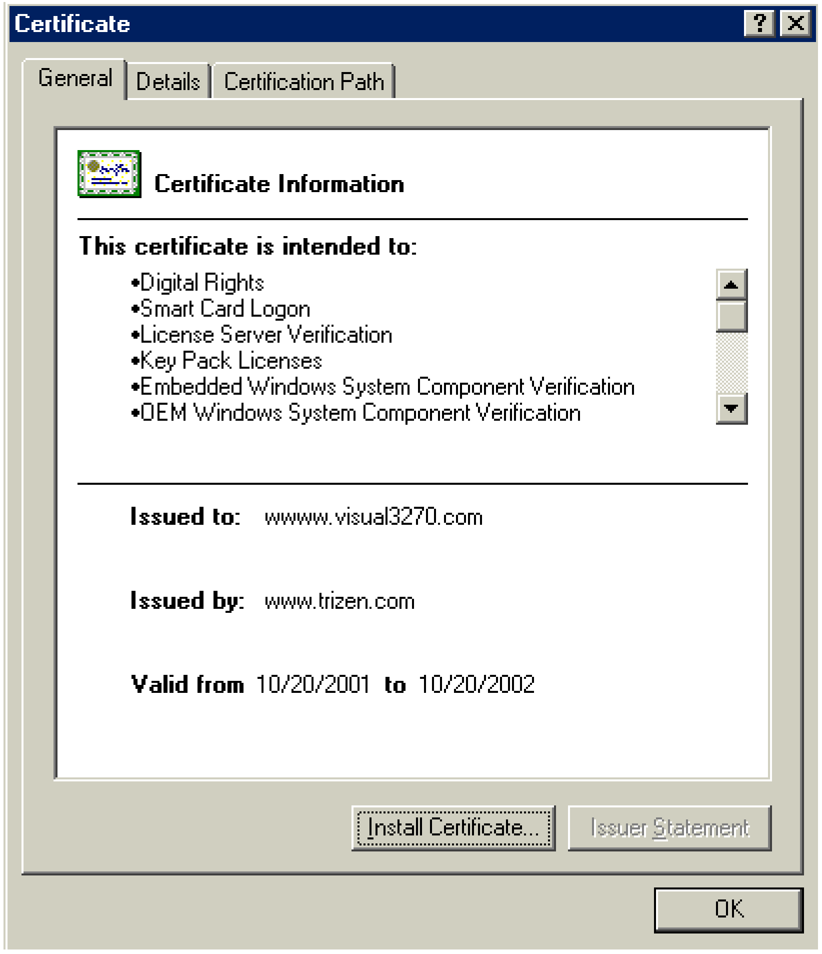
Issued to: www.trizen.com

Issued by: www.trizen.com

Valid from 10/20/2001 **to** 10/18/2011

In the above image, Windows now has some information to work with, unlike the first time we double clicked on the CA Certificate. If you then click on the Certification Path tab at the top you will see that Windows now says the certificate path is Ok as well.

At this point create and sign a certificate, and then double click on the .crt file of the newly created signed certificate in the *c:\CARoot\newcerts* directory (you may have to change the name to have a .crt extension and also remove the header information). If you followed the directions in the **Creating a New Public Certificate and Key Pair** then you already have a certificate that you can simply double click on (making sure you made it compatible for windows). You should see the following dialog:



Certificate Information

This certificate is intended to:

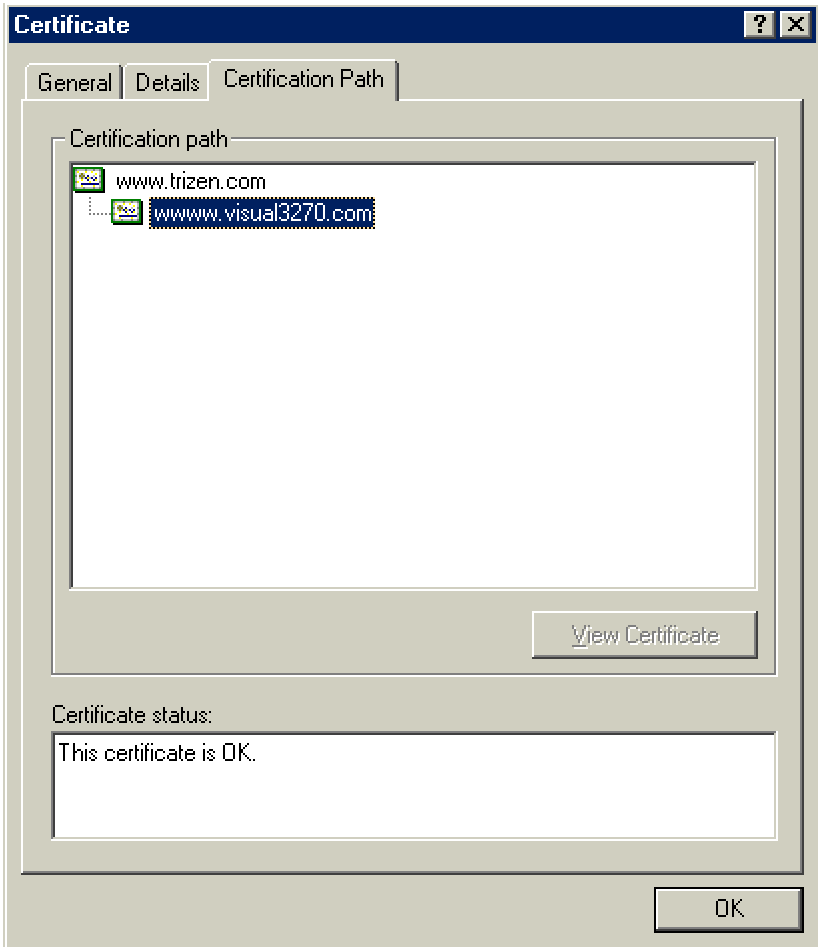
- •Digital Rights
- •Smart Card Logon
- •License Server Verification
- •Key Pack Licenses
- •Embedded Windows System Component Verification
- •OEM Windows System Component Verification

Issued to: wwww.visual3270.com

Issued by: www.trizen.com

Valid from 10/20/2001 to 10/20/2002

Notice that you have not installed this certificate. It is already trusted, because it's root certificate is trusted which is the CA Certificate that was installed earlier. If you click on the *Certification Path* tab you will see the certificate chain:



Certification path

www.trizen.com
wwww.visual3270.com

Certificate status:

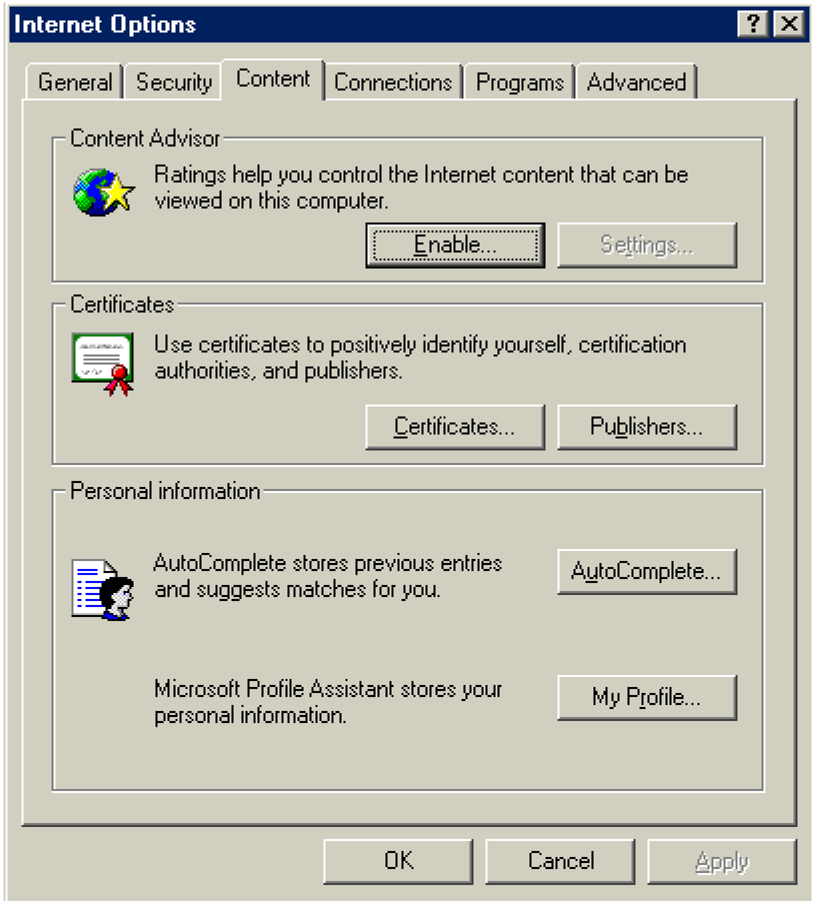
This certificate is OK.

Notice that the Root certificate for this newly created and signed certificate is the www.trizen.com (Common Name) certificate. If you click on this, then you can view the CA certificate.

Using Internet Explorer to Install the Certificate

An alternate way to install certificates is to use Internet Explorer which will allow you to install .pem files (assuming they have been modified to conform to Windows parsing) or any other files that conform to PEM or x509 structure.

Open up Internet Explorer and from the menu bar select Tools | Internet Options. When you select the Content tab you should see the following:


Internet Options

Content Advisor

Ratings help you control the Internet content that can be viewed on this computer.

Certificates

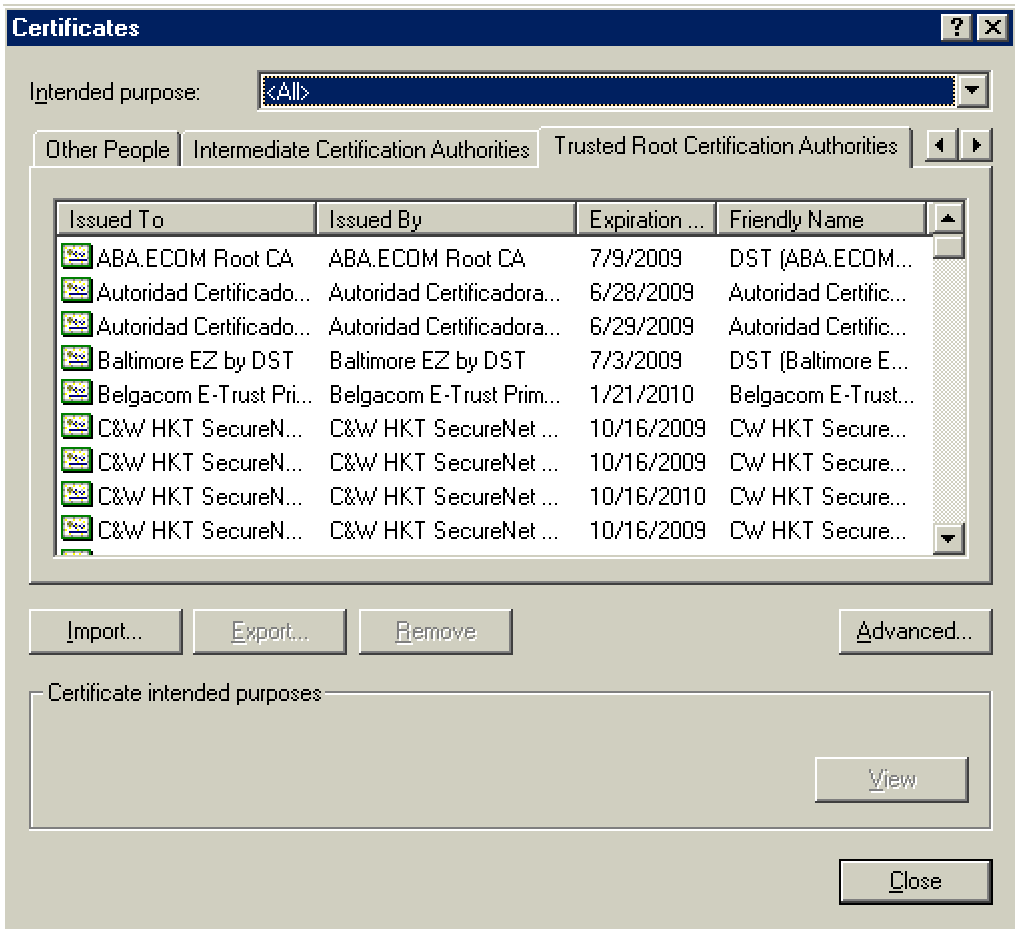
Use certificates to positively identify yourself, certification authorities, and publishers.

Personal information

AutoComplete stores previous entries and suggests matches for you.

Microsoft Profile Assistant stores your personal information.

When the above dialog appears (assuming you're using Internet 5.0) click on the *Certificates…* button. You should get the following dialog:


Certificates
Intended purpose:
Other People
Intermediate Certification Authorities
Trusted Root Certification Authorities
Import...
Export...
Remove
Advanced...
Certificate intended purposes
View
Close

Intended purpose: <All>

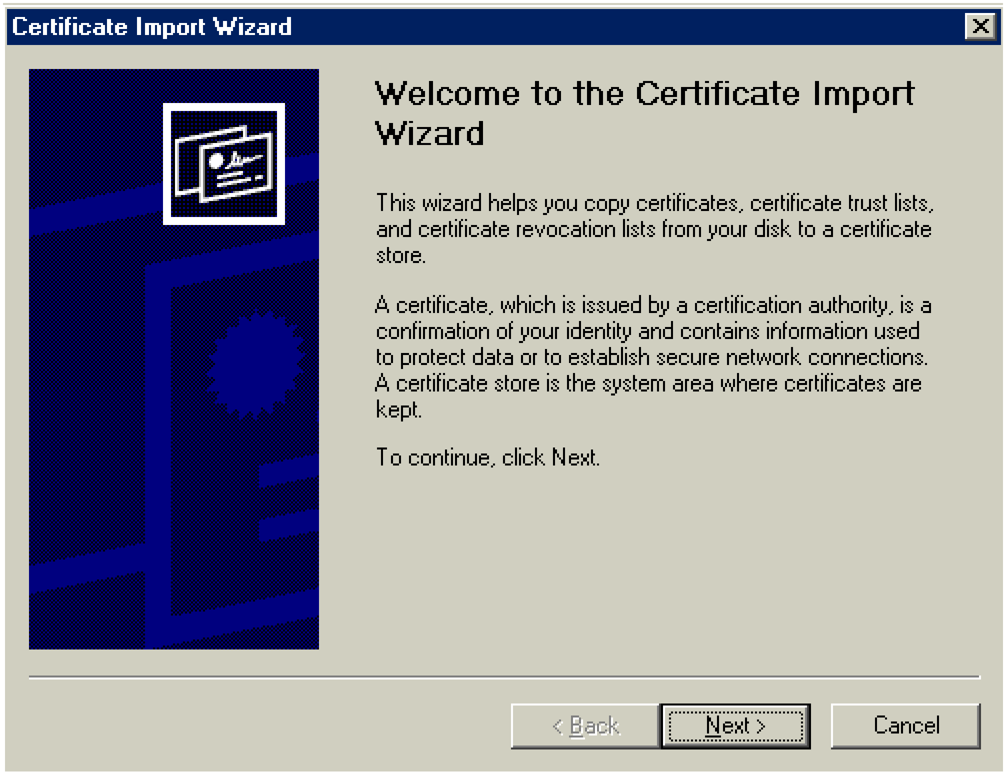
Issued To	Issued By	Expiration ...	Friendly Name
ABA.ECOM Root CA	ABA.ECOM Root CA	7/9/2009	DST (ABA.ECOM...
Autoridad Certificado...	Autoridad Certificadora...	6/28/2009	Autoridad Certific...
Autoridad Certificado...	Autoridad Certificadora...	6/29/2009	Autoridad Certific...
Baltimore EZ by DST	Baltimore EZ by DST	7/3/2009	DST (Baltimore E...
Belgacom E-Trust Pri...	Belgacom E-Trust Prim...	1/21/2010	Belgacom E-Trust...
C&W HKT SecureN...	C&W HKT SecureNet ...	10/16/2009	CW HKT Secure...
C&W HKT SecureN...	C&W HKT SecureNet ...	10/16/2009	CW HKT Secure...
C&W HKT SecureN...	C&W HKT SecureNet ...	10/16/2010	CW HKT Secure...
C&W HKT SecureN...	C&W HKT SecureNet ...	10/16/2009	CW HKT Secure...

Certificate Chaining

From here you can see all of the Trusted Root Certificates (Certificate Authorities). If one of these CA has signed a certificate that your computer is validating, the root will be accepted. The root certificate is at the top of the chain for any signed certificates. This goes hand in hand with certificate chaining, whereas there can be many Certificate Authorities, each one positioning itself in the certificate chain. In this case Trizen is the Root certificate and any certificates we sign will have a depth of 1 (starting at 0) where position 0 is the signed certificate and position 1 is the Trizen Root. If you are developing secure applications using Visual SSL or OpenSSL then you should review the material on client verification and certificate depths.

Importing the CA Certificate

On the above dialog, press the *Import...* button. You should see the following dialog:



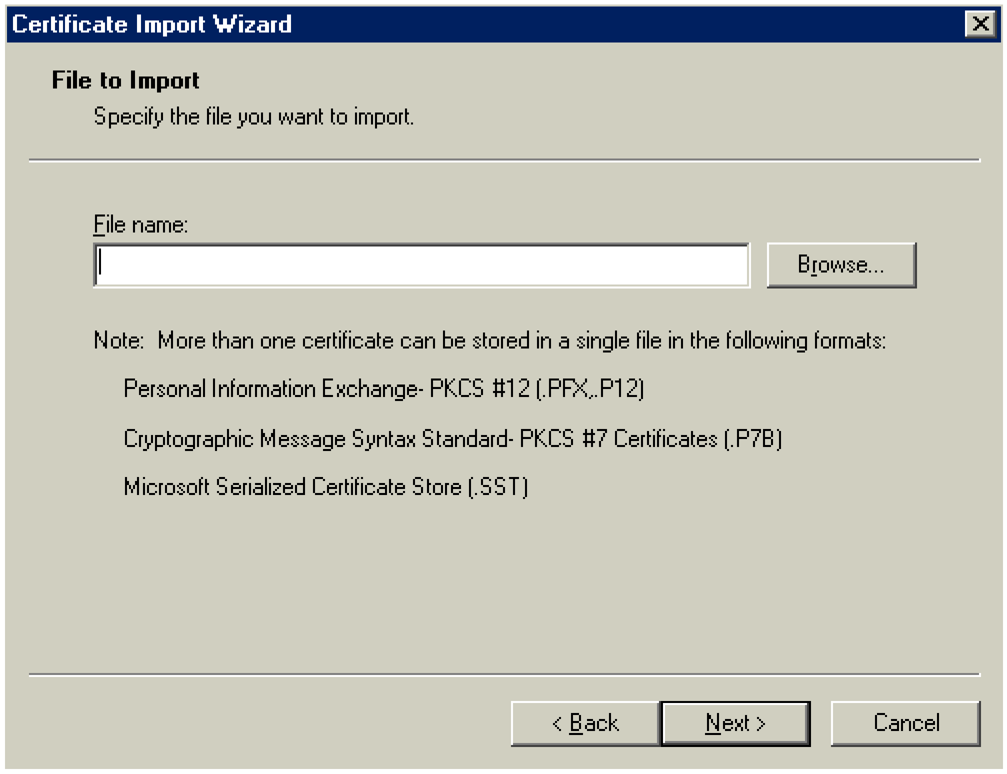
Welcome to the Certificate Import Wizard

This wizard helps you copy certificates, certificate trust lists, and certificate revocation lists from your disk to a certificate store.

A certificate, which is issued by a certification authority, is a confirmation of your identity and contains information used to protect data or to establish secure network connections. A certificate store is the system area where certificates are kept.

To continue, click Next.

Just press the *Next >* button. You should see the next dialog:



File to Import

Specify the file you want to import.

File name:

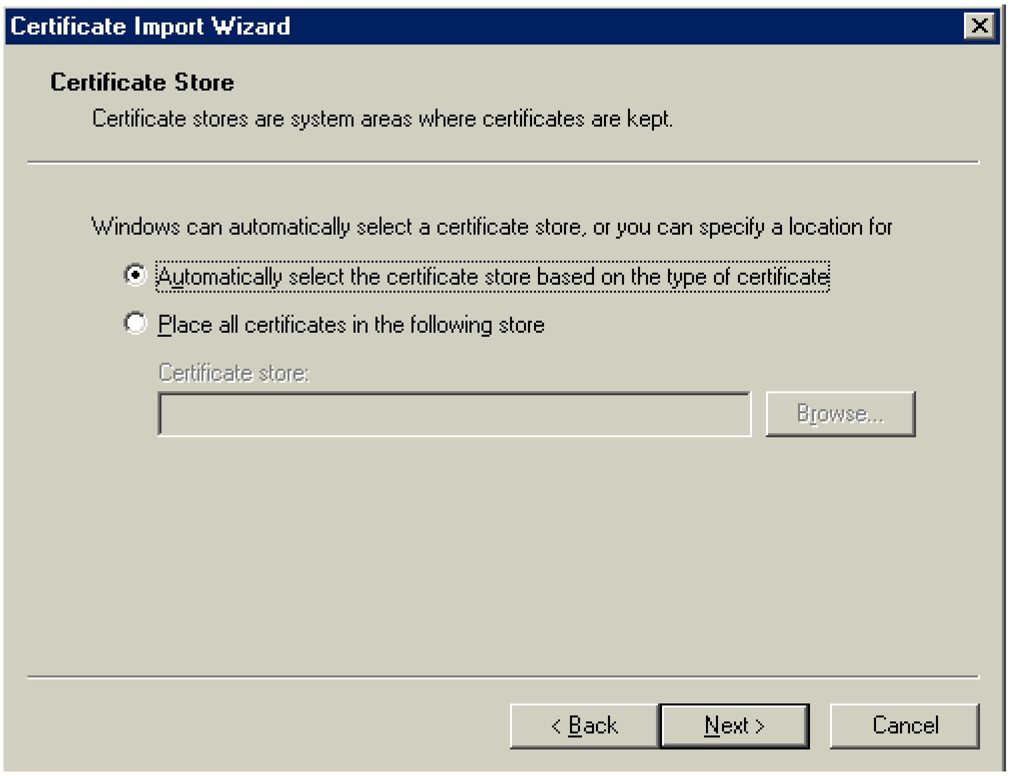
Note: More than one certificate can be stored in a single file in the following formats:

Personal Information Exchange- PKCS #12 (.PFX,.P12)

Cryptographic Message Syntax Standard- PKCS #7 Certificates (.P7B)

Microsoft Serialized Certificate Store (.SST)

Press the *Browse…* button and navigate to the signed certificate you wish to import *i.e. ca.crt* in the *private* directory of the CARoot folder. You may have to view *.* file types. Press the *Next* > button to continue.



Certificate Import Wizard

Certificate Store

Certificate stores are system areas where certificates are kept.

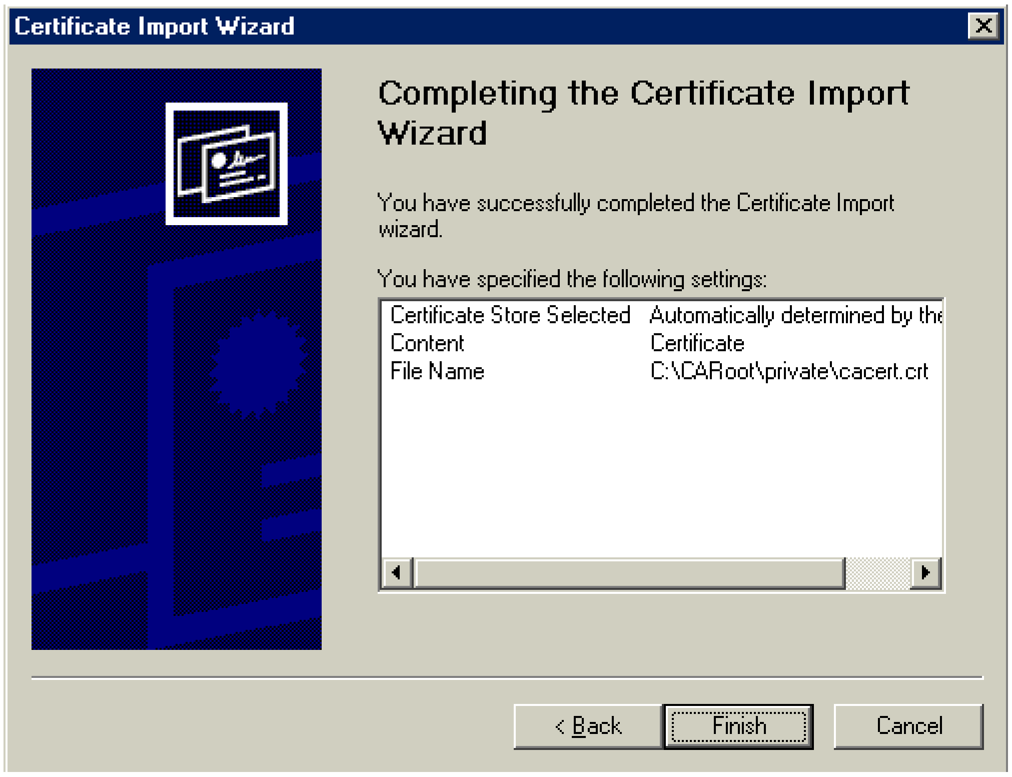
Windows can automatically select a certificate store, or you can specify a location for

(●) Automatically select the certificate store based on the type of certificate

( ) Place all certificates in the following store

Certificate store:

You should just accept the setting in the above dialog and press the *Next* > button. You should see the following dialog.



Completing the Certificate Import Wizard

You have successfully completed the Certificate Import wizard.

You have specified the following settings:

Certificate Store Selected	Automatically determined by the
Content	Certificate
File Name	C:\CARoot\private\cacert.crt

At this point it is the same as if you had double clicked on a .crt file. Remember you can't double click on .pem files and get this dialog, you must either go through Internet Explorer or change the name so that it has a .crt extension—making sure you have removed all the header information and leaving only the ------- BEGIN CERTIFICATE ----- and the -------- END CERTIFICATE ------- data.

Using the OpenSSL Applications s_server.exe & s_client.exe

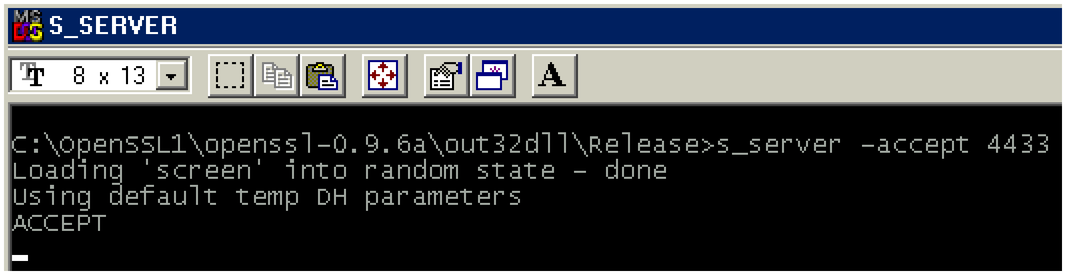
Undoubtedly you will want to test your certificates. OpenSSL provides you with a rich set of applications that can be found in the *c:\OpenSSL\openssl-0.9.6a\out32dll\release* directory. We have already discussed several of them in the above sections, but as a developer you will want to use some of the other applications to test your own SSL enabled applications. Remember that these utilities are all MS-DOS based and it is recommended that you place a copy of command.com or cmd.exe (for NT) in the *c:\OpenSSL\openssl-0.9.6a\out32dll\release* directory.

Running s_client.exe and s_server.exe

We can guarantee that you will use s_server.exe at some point during your SSL development life cycle. This is a great utility that will test your client's capabilities and ensure that you are communicating properly. If your client can communicate with s_server.exe then you can be pretty sure it will perform accurately and appropriately with other SSL enabled servers including https://www.verisign.com and others.

Copy the *server.pem* and *client.pem* files from the C:\OpensSSL\openssl-0.9.6a\apps directory to the C:\OpenSSL\openssl-0.9.6a\out32dll\release directory.
Double click on your command.com or cmd.exe file in the *release* directory. When the command window pops up, type the following at the command prompt:

C:\OpenSSL\openssl-0.9.6a\out32dll\release>*s_server –accept 4433*



```
C:\OpenSSL1\openssl-0.9.6a\out32dll\Release>s_server -accept 4433
Loading 'screen' into random state - done
Using default temp DH parameters
ACCEPT
```

The s_server.exe application is now ready to accept clients. At this point, stop the server by pressing Ctrl+C twice to exit the application. Type in the following at the command prompt and press enter:

C:\OpenSSL\openssl-0.9.6a\out32dll\release>*s_server -help*

This will bring up all of the options that can be utilized with the s_server.exe application. If you use none of the options the server will accept on port 4433 using server.pem as the certificate which will also contain the private key. Note that you can place the private key into the same file as the public certificate when using OpenSSL.

If you haven’t already stopped the execution of the s_server.exe application, just press Ctrl and the letter C twice while still holding the Control key. At this point copy the ca.key and the ca.crt file to this *release* directory (from CARoot\private). Then at the command prompt type the following:

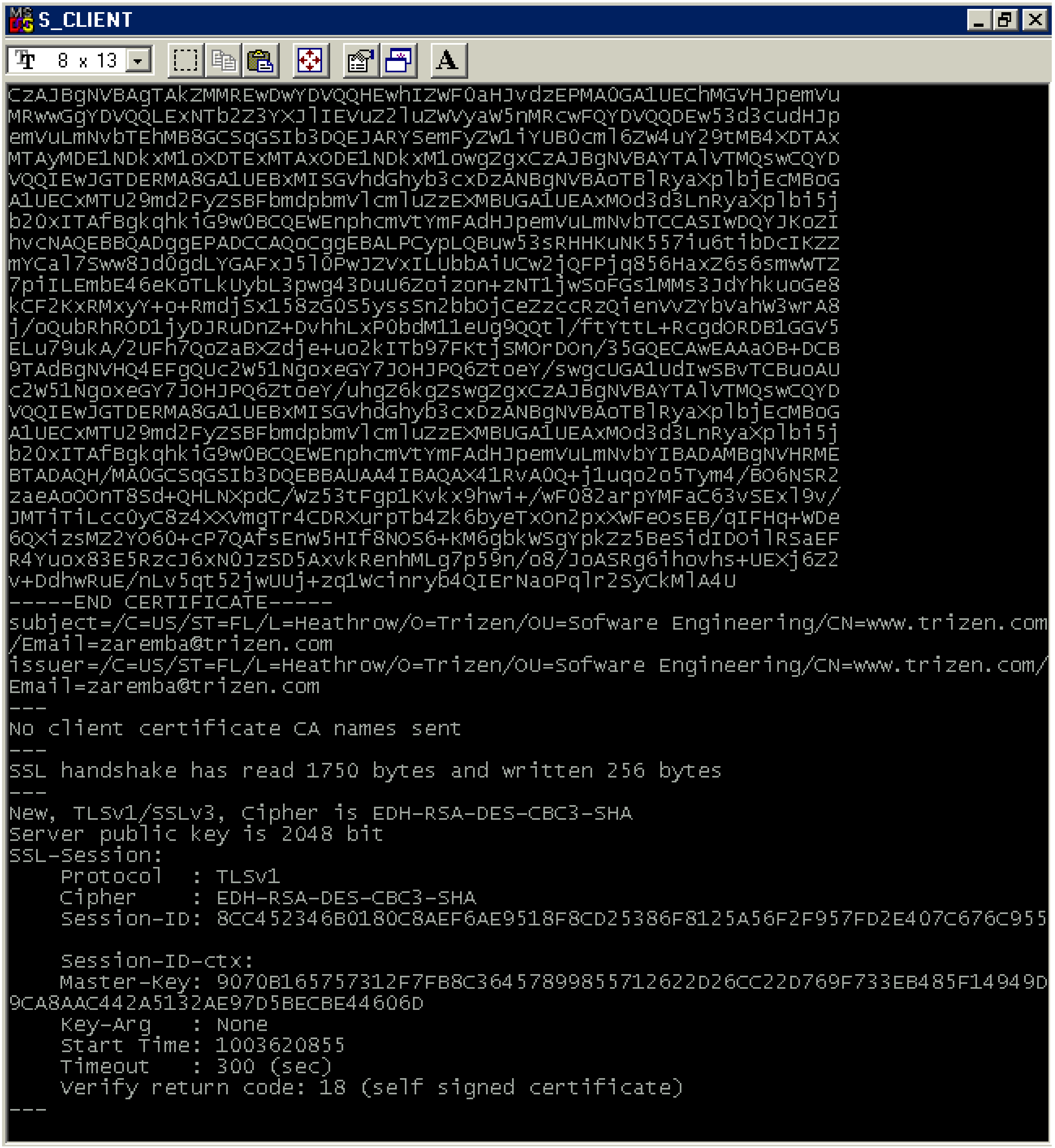
```
C:\OpenSSL\openssl-0.9.6a\out32dll\release>s_server –cert ca.crt –key ca.key
```

This will start the s_server application with the CA Public Certificate and CA Private Key files that were created earlier. If it asks for the PEM password, just type in the password that you made when you created the CA Key file.

At this point, let the s_server.exe continue to run and double click on the command.com or cmd.exe (for NT) file to bring up a new MS-DOS prompt. At the DOS prompt type the following:

```
C:\OpenSSL\openssl-0.9.6a\out32dll\release>s_client–cert ca.crt –key ca.key
```

You should see the following:



```
CzAJBgNVBAgTAkZMMREwDwYDVQQHEwhIZWF0aHJvdzEPMA0GA1UEChMGVHJpemVu
MRwwGgYDVQQLExNTb2Z3YXJlIEVuZ2luZWVyaW5nMRcwFQYDVQQDEw53d3cudHJp
emVuLmNvbTEhMB8GCSqGSIb3DQEJARYSemFyZW1iYUB0cml6ZW4uY29tMB4XDTAx
MTAyMDE1NDkxM1oXDTExMTAxODE1NDkxM1owgZgxCzAJBgNVBAYTAlVTMQswCQYD
VQQIEwJGTDERMA8GA1UEBxMISGVhdGhyb3cxDzANBgNVBAoTBlRyaXplbjEcMBoG
A1UECxMTU29md2FyZSBFbmdpbmVlcmluZzEXMBUGA1UEAxMOd3d3LnRyaXplbi5j
b20xITAfBgkqhkiG9w0BCQEWEnphcmVtYmFAdHJpemVuLmNvbTCCASIwDQYJKoZI
hvcNAQEBBQADggEPADCCAQoCggEBALPCypLQBuw53sRHHKuNK557iu6tibDcIKZZ
mYCal7Sww8Jd0gdLYGAFxJ5l0PwJZVxILUbbAiUCw2jQFPjq856HaxZ6s6smwWTZ
7piILEmbE46eKoTLkUybL3pwg43DuU6Zoizon+zNT1jwSoFGs1MMs3JdYhkuoGe8
kCF2KxRMxyY+o+RmdjSx158zG0S5ysssn2bbojCeZzccRzQienVvZYbvahw3wrA8
j/oQubRhROD1jyDJRuDnZ+DvhhLxP0bdM11eUg9QQtl/ftYttL+RcgdORDB1GGV5
ELu79ukA/2UFh7QoZaBXZdje+uo2kITb97FKtjSMOrDOn/35GQECAwEAAaOB+DCB
9TAdBgNVHQ4EFgQUc2W51NgoxeGY7JOHJPQ6ZtoeY/swgcUGA1UdIwSBvTCBuoAU
c2W51NgoxeGY7JOHJPQ6ZtoeY/uhgZ6kgZswgZgxCzAJBgNVBAYTAlVTMQswCQYD
VQQIEwJGTDERMA8GA1UEBxMISGVhdGhyb3cxDzANBgNVBAoTBlRyaXplbjEcMBoG
A1UECxMTU29md2FyZSBFbmdpbmVlcmluZzEXMBUGA1UEAxMOd3d3LnRyaXplbi5j
b20xITAfBgkqhkiG9w0BCQEWEnphcmVtYmFAdHJpemVuLmNvbYIBADAMBgNVHRME
BTADAQH/MA0GCSqGSIb3DQEBBAUAA4IBAQAX41RvA0Q+j1uqo2o5Tym4/BO6NSR2
zaeAoOOnT8Sd+QHLNXpdC/Wz53tFgp1Kvkx9hwi+/wF082arpYMFaC63vSEx19v/
JMTiTiLcc0yC8z4XXVmgTr4CDRXurpTb4Zk6byeTxOn2pxXWFeOsEB/qIFHq+WDe
6QXizsMZ2YO60+cP7QAfsEnW5HIf8NOS6+KM6gbkWSgYpkZz5BeSidIDOilRSaEF
R4Yuox83E5RzcJ6xN0JzSD5AxvkRenhMLg7p59n/o8/JoASRg6ihovhs+UEXj6Z2
v+DdhwRuE/nLv5qt52jwUUj+zq1WcinryB4QIErNaoPqlr2SyCkMlA4U
-----END CERTIFICATE-----
subject=/C=US/ST=FL/L=Heathrow/O=Trizen/OU=Sofware Engineering/CN=www.trizen.com
/Email=zaremba@trizen.com
issuer=/C=US/ST=FL/L=Heathrow/O=Trizen/OU=Sofware Engineering/CN=www.trizen.com/
Email=zaremba@trizen.com
---
No client certificate CA names sent
---
SSL handshake has read 1750 bytes and written 256 bytes
---
New, TLSv1/SSLv3, Cipher is EDH-RSA-DES-CBC3-SHA
Server public key is 2048 bit
SSL-Session:
    Protocol  : TLSv1
    Cipher    : EDH-RSA-DES-CBC3-SHA
    Session-ID: 8CC452346B0180C8AEF6AE9518F8CD25386F8125A56F2F957FD2E407C676C955

    Session-ID-ctx:
    Master-Key: 9070B165757312F7FB8C364578998557126 22D26CC22D769F733EB485F14949D
9CA8AAC442A5132AE97D5BECBE44606D
    Key-Arg   : None
    Start Time: 1003620855
    Timeout   : 300 (sec)
    Verify return code: 18 (self signed certificate)
---
```

If you did not see this make sure you have placed the *ca.crt* and *ca.key* file into the same directory as the *s_client.exe* application (c:\OpenSSL\openssl-0.9.6a\out32dll\release). Also, you may want to add the switch *–connect 127.0.0.1:4433* which represents the host IP and Port number of the *s_server.exe* application.

Notice the last line: Verify return code: 18 (self signed certificate). Which makes sense, the s_client.exe application will automatically attempt to verify the server's certificate, which in the case of the s_server.exe application we are using the CA certificate which was self signed! At this point shutdown your s_client.exe application and then the s_server.exe application.

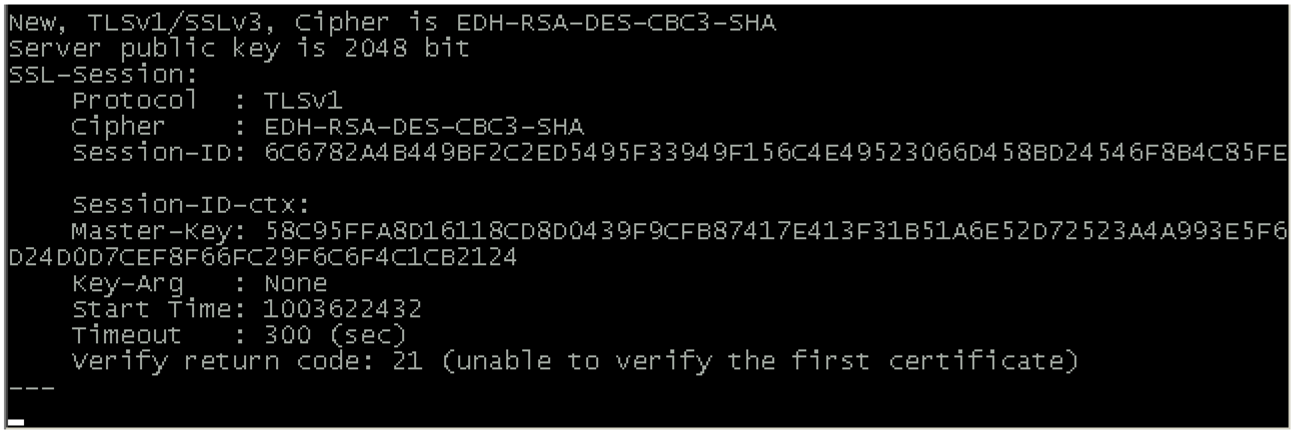
Copy the signed certificate signedcert.crt created earlier or *01.pem* file (or whatever it is named in the *C:\CARoot* directory) to the *C:\OpenSSL\openssl-0.9.6a\out32dll\release* directory. Copy the cert.key which was used to create the request certificate, then later signed —this is the public key pair (actually there are three, a publicly signed certificate, a request certificate and a key file). Type in the following into one of the MD-DOS window command prompts:

C:\OpenSSL\openssl-0.9.6a\out32dll\release>*s_server –cert signedcert.crt –key cert.key*

Your server should be up and running. At another MS-DOS prompt (open another window if necessary) type in the following:

C:\OpenSSL\openssl-0.9.6a\out32dll\release>*s_client –cert signedcert.crt –key cert.key*

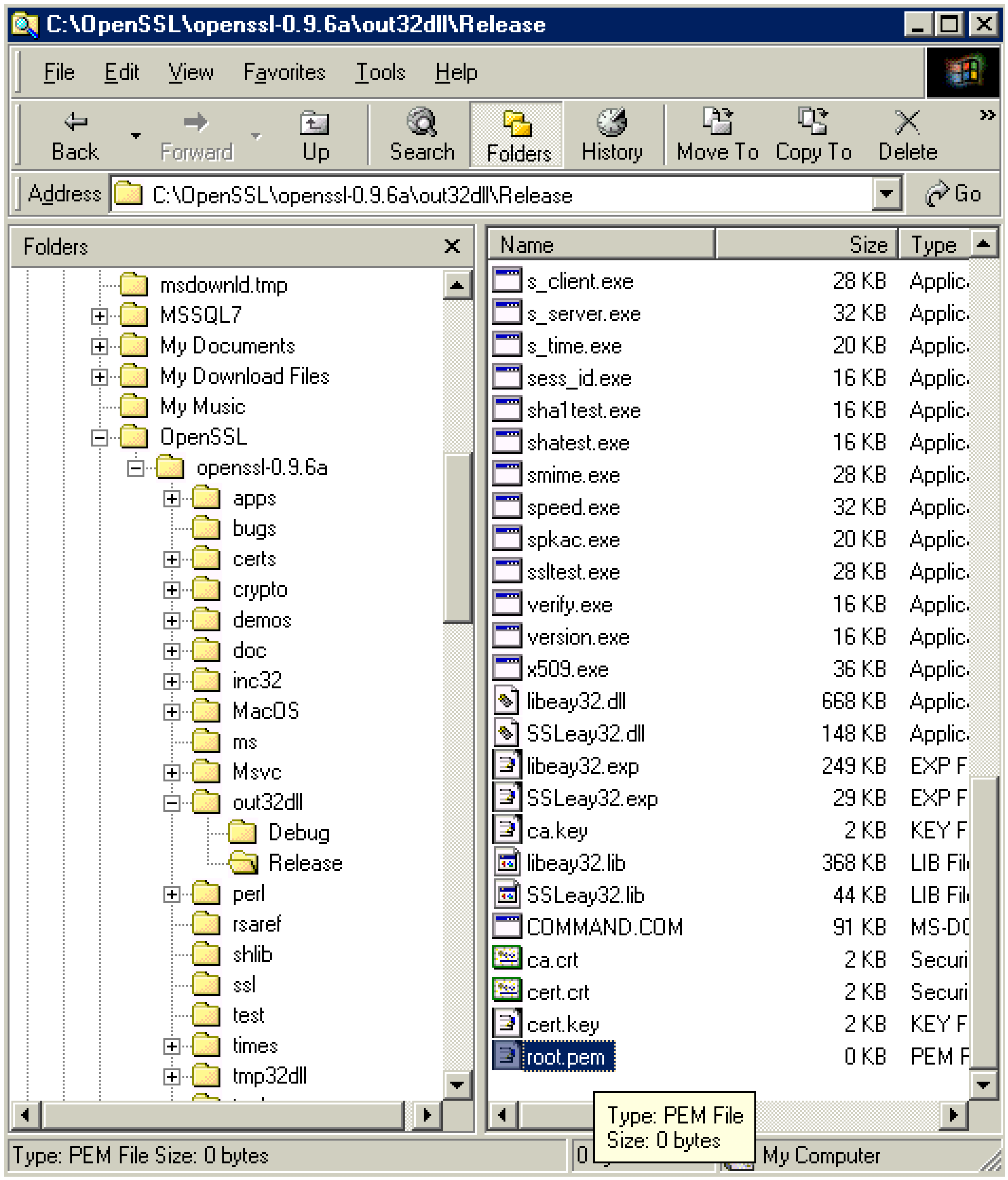
If all has gone well then you should get another verification error? At least you're not getting the self signed certificate error "18" but is error "21" any better? This error represents the problem of not being able to validate a server's certificate using a Trusted Certificate Store (such as that created when installing certificates in Windows above).


```
New, TLSv1/SSLv3, Cipher is EDH-RSA-DES-CBC3-SHA
Server public key is 2048 bit
SSL-Session:
    Protocol  : TLSv1
    Cipher    : EDH-RSA-DES-CBC3-SHA
    Session-ID: 6C6782A4B449BF2C2ED5495F33949F156C4E49523066D458BD24546F8B4C85FE

    Session-ID-ctx:
    Master-Key: 58C95FFA8D16118CD8D0439F9CFB87417E413F31B51A6E52D72523A4A993E5F6
D24D0D7CEF8F66FC29F6C6F4C1CB2124
    Key-Arg   : None
    Start Time: 1003622432
    Timeout   : 300 (sec)
    Verify return code: 21 (unable to verify the first certificate)
---
```

This has everything to do with the client's verification process. Basically we need to inform the s_client.exe application that it should look in a file for those certificates that it should trust! This is accomplished by using the –CAFile switch. Stop running the s_client.exe application but **keep the s_server application running**.

Creating a CAFile

A CAFile is very easy to create and consists of nothing more than a bunch of public certificates appended to each other. To create a CAFile, from windows right mouse click on the Start button. Select the Explore pop up menu item. When the explorer window pops up navigate to *c:\OpenSSL\openssl-0.9.6a\out32dll\release*. In the right hand pane, right mouse click in the area where the release files are located, selecting New | Text Document. It should now appear in the right pane awaiting for you to give it a name. Rename the file root.pem. Your directory should look like the following:



C:\OpenSSL\openssl-0.9.6a\out32dll\Release

Address C:\OpenSSL\openssl-0.9.6a\out32dll\Release

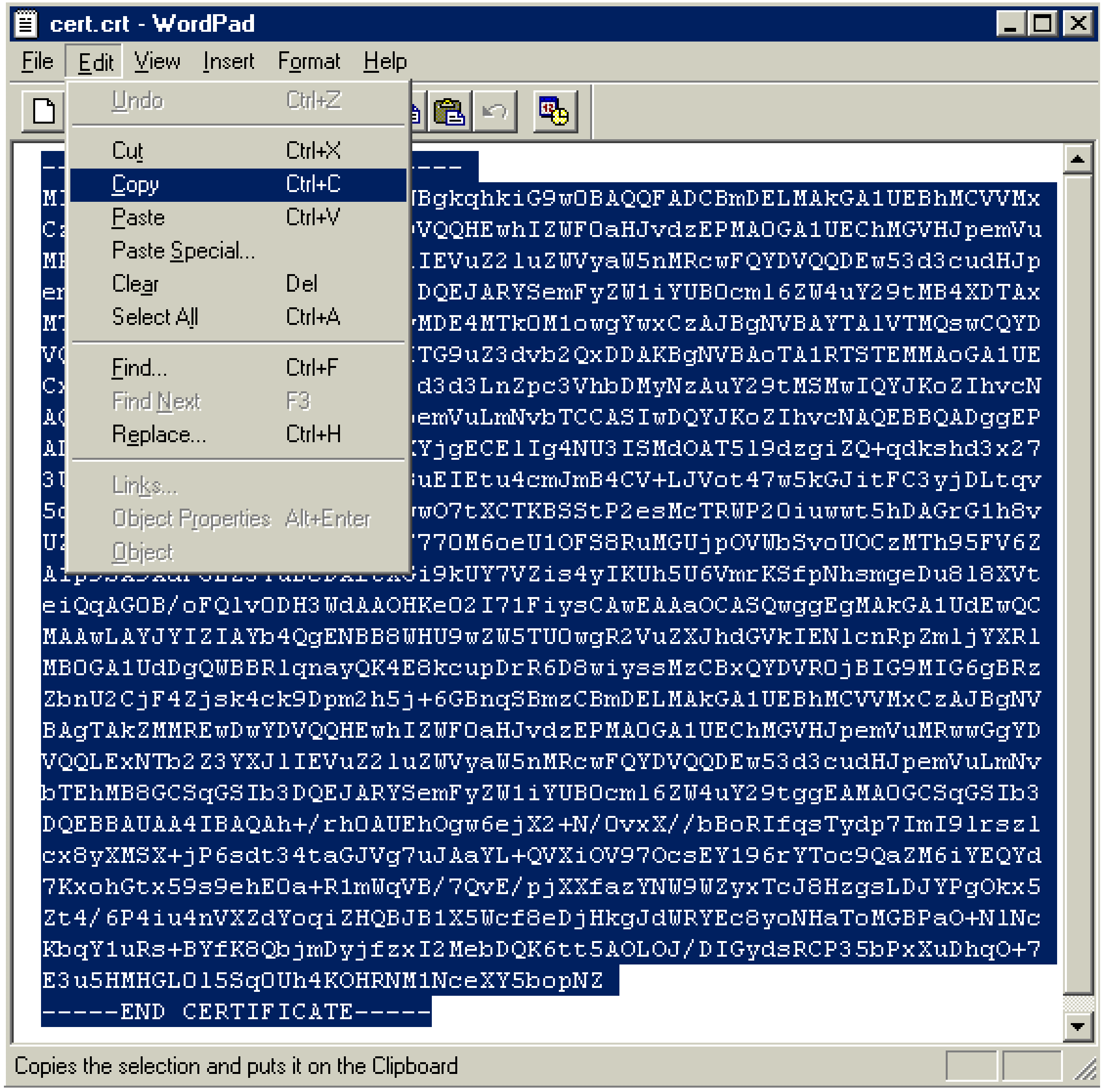
```
msdownld.tmp
MSSQL7
My Documents
My Download Files
My Music
OpenSSL
  openssl-0.9.6a
    apps
    bugs
    certs
    crypto
    demos
    doc
    inc32
    MacOS
    ms
    Msvc
    out32dll
      Debug
      Release
    perl
    rsaref
    shlib
    ssl
    test
    times
    tmp32dll
```

Name	Size	Type
s_client.exe	28 KB	Applic
s_server.exe	32 KB	Applic
s_time.exe	20 KB	Applic
sess_id.exe	16 KB	Applic
sha1test.exe	16 KB	Applic
shatest.exe	16 KB	Applic
smime.exe	28 KB	Applic
speed.exe	32 KB	Applic
spkac.exe	20 KB	Applic
ssltest.exe	28 KB	Applic
verify.exe	16 KB	Applic
version.exe	16 KB	Applic
x509.exe	36 KB	Applic
libeay32.dll	668 KB	Applic
SSLeay32.dll	148 KB	Applic
libeay32.exp	249 KB	EXP F
SSLeay32.exp	29 KB	EXP F
ca.key	2 KB	KEY F
libeay32.lib	368 KB	LIB Fil
SSLeay32.lib	44 KB	LIB Fil
COMMAND.COM	91 KB	MS-D
ca.crt	2 KB	Securi
cert.crt	2 KB	Securi
cert.key	2 KB	KEY F
root.pem	0 KB	PEM F

Type: PEM File
Size: 0 bytes

Type: PEM File Size: 0 bytes | 0 | My Computer

Double click the new root.pem file to open up the text editor. You should have a blank file. Next, right mouse click on the signedcert.crt file and select Open With and either select WordPad if its available or select Choose Program and use the WordPad program. When WordPad opens up, select all the text (Edit | Select All) in the signedcert.crt file and then copy the selection to the clipboard (Edit | Copy):



```
--                                 -----
MI                                 IBgkqhkiG9w0BAQQFADCBmDELMAkGA1UEBhMCVVMx
C                                  VQQHEwhIZWF0aHJvdzEPMA0GA1UEChMGVHJpemVu
MB                                 IEVuZ2luZWVyaW5nMRcwFQYDVQQDEw53d3cudHJp
er                                 DQEJARYSemFyZW1iYUB0cml6ZW4uY29tMB4XDTAx
MT                                 MDE4MTk0M1owgYwxCzAJBgNVBAYTAlVTMQswCQYD
VQ                                 TG9uZ3dvb2QxDDAKBgNVBAoTA1RTSTEMMAoGA1UE
Cx                                 d3d3LnZpc3VhbDMyNzAuY29tMSMwIQYJKoZIhvcN
AQ                                 emVuLmNvbTCCASIwDQYJKoZIhvcNAQEBBQADggEP
AD                                 YjgECElIg4NU3ISMdOAT5l9dzgiZQ+qdkshd3x27
3U                                 uEIEtu4cmJmB4CV+LJVot47w5kGJitFC3yjDLtqv
5c                                 wO7tXCTKBSStP2esMcTRWP2Oiuwwt5hDAGrG1h8v
U2                                 770M6oeU1OFS8RuMGUjpOVWbSvoUOCzMTh95FV6Z
A                                  i9kUY7VZis4yIKUh5U6VmrKSfpNhsmgeDu8l8XVt
eiQqAGOB/oFQlvODH3WdAAOHKeO2I71FiysCAwEAAaOCASQwggEgMAkGA1UdEwQC
MAAwLAYJYIZIAYb4QgENBB8WHU9wZW5TUOwgR2VuZXJhdGVkIENlcnRpZmljYXRl
MB0GA1UdDgQWBBRlqnayQK4E8kcupDrR6D8wiyssMzCBxQYDVR0jBIG9MIG6gBRz
ZbnU2CjF4Zjsk4ck9Dpm2h5j+6GBnqSBmzCBmDELMAkGA1UEBhMCVVMxCzAJBgNV
BAgTAkZMMREwDwYDVQQHEwhIZWF0aHJvdzEPMA0GA1UEChMGVHJpemVuMRwwGgYD
VQQLExNTb2Z3YXJlIEVuZ2luZWVyaW5nMRcwFQYDVQQDEw53d3cudHJpemVuLmNv
bTEhMB8GCSqGSIb3DQEJARYSemFyZW1iYUB0cml6ZW4uY29tggEAMA0GCSqGSIb3
DQEBBAUAA4IBAQAh+/rhOAUEhOgw6ejX2+N/OvxX//bBoRIfqsTydp7ImI9lrszl
cx8yXMSX+jP6sdt34taGJVg7uJAaYL+QVXiOV97OcsEY196rYToc9QaZM6iYEQYd
7KxohGtx59s9ehE0a+R1mWqVB/7QvE/pjXXfazYNW9WZyxTcJ8HzgsLDJYPgOkx5
Zt4/6P4iu4nVXZdYoqiZHQBJB1X5Wcf8eDjHkgJdWRYEc8yoNHaToMGBPaO+NlNc
KbqY1uRs+BYfK8QbjmDyjfzxI2MebDQK6tt5AOLOJ/DIGydsRCP35bPxXuDhqO+7
E3u5HMHGL0l5SqOUh4KOHRNM1NceXY5bopNZ
-----END CERTIFICATE-----
```

Now go to the other WordPad application which represents the blank root.pem file and paste this data into that file then save the file and exit. You have successfully created a CAFile. If you have not already closed the s_client.exe application do so now, then at the command prompt type the following:

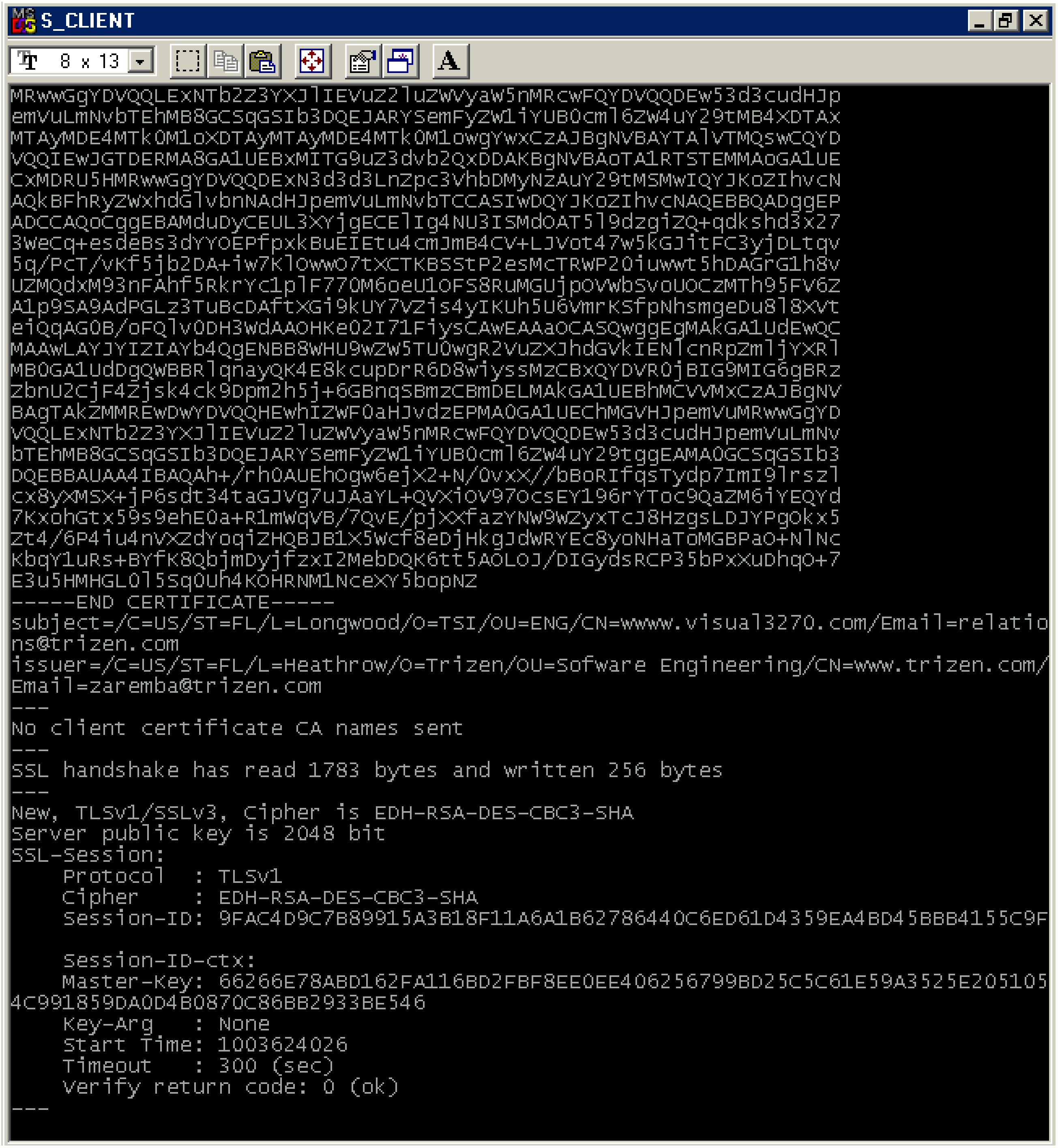
C:\OpenSSL\openssl-0.9.6a\out32dll\release>*s_client –cert signedcert.crt –key cert.key –CAFile root.pem*

If all goes well, you should see the same error! "Unable to verify the first certificate" is not what you would probably expect, but we have only copied the signed certificate, we still have to copy the CA certificate (ca.crt). Shutdown the s_client.exe application and copy the ca.crt certificate data to the bottom of the root.pem file using the same method as above. The root.pem file should now look something similar to the following:

```
-----BEGIN CERTIFICATE-----
MIIExzCCA6+gAwIBAgIBBzANBgkqhkiG9w0BAQQFADCBmDELMAkGA1UEBhMCVVMx
CzAJBgNVBAgTAkZMMREwDwYDVQQHEwhIZWF0aHJvdzEPMA0GA1UEChMGVHJpemVu
MRwwGgYDVQQLExNTb2Z3YXJlIEVuZ2luZWVyaW5nMRcwFQYDVQQDEw53d3cudHJp
emVuLmNvbTEhMB8GCSqGSIb3DQEJARYSemFyZW1iYUB0cml6ZW4uY29tMB4XDTAx
MTAyMDE4MTk0M1oXDTAyMTAyMDE4MTk0M1owgYwxCzAJBgNVBAYTAlVTMQswCQYD
VQQIEwJGTDERMA8GA1UEBxMITG9uZ3dvb2QxDDAKBgNVBAoTA1RTSTEMMAoGA1UE
CxMDRU5HMRwwGgYDVQQDExN3d3d3LnZpc3VhbDMyNzAuY29tMSMwIQYJKoZIhvcN
AQkBFhRyZWxhdGlvbnNAdHJpemVuLmNvbTCCASIwDQYJKoZIhvcNAQEBBQADggEP
ADCCAQoCggEBAMduDyCEUL3XYjgECElIg4NU3ISMdOAT5l9dzgiZQ+qdkshd3x27
3WeCq+esdeBs3dYYOEPfpxkBuEIEtu4cmJmB4CV+LJVot47w5kGJitFC3yjDLtqv
5q/PcT/vKf5jb2DA+iw7KlOwwO7tXCTKBSStP2esMcTRWP20iuwwt5hDAGrG1h8v
UZMQdxM93nFAhf5RkrYc1plF770M6oeU1OFS8RuMGUjpOVWbSvoUOCzMTh95FV6Z
A1p9SA9AdPGLz3TuBcDAftXGi9kUY7VZis4yIKUh5U6VmrKSfpNhsmgeDu8l8XVt
eiQqAG0B/oFQlv0DH3WdAAOHKe02I71FiysCAwEAAaOCASQwggEgMAkGA1UdEwQC
MAAwLAYJYIZIAYb4QgENBB8WHU9wZW5TU0wgR2VuZXJhdGVkIENlcnRpZmljYXRl
MB0GA1UdDgQWBBRlqnayQK4E8kcupDrR6D8wiyssMzCBxQYDVR0jBIG9MIG6gBRz
ZbnU2CjF4Zjsk4ck9Dpm2h5j+6GBnqSBmzCBmDELMAkGA1UEBhMCVVMxCzAJBgNV
BAgTAkZMMREwDwYDVQQHEwhIZWF0aHJvdzEPMA0GA1UEChMGVHJpemVuMRwwGgYD
VQQLExNTb2Z3YXJlIEVuZ2luZWVyaW5nMRcwFQYDVQQDEw53d3cudHJpemVuLmNv
bTEhMB8GCSqGSIb3DQEJARYSemFyZW1iYUB0cml6ZW4uY29tggEAMA0GCSqGSIb3
DQEBBAUAA4IBAQAh+/rh0AUEhOgw6ejX2+N/0vxX//bBoRIfqsTydp7ImI9lrszl
cx8yXMSX+jP6sdt34taGJVg7uJAaYL+QVXiOV97OcsEY196rYToc9QaZM6iYEQYd
7KxohGtx59s9ehE0a+R1mWqVB/7QvE/pjXXfazYNW9WZyxTcJ8HzgsLDJYPgOkx5
Zt4/6P4iu4nVXZdYoqiZHQBJB1X5Wcf8eDjHkgJdWRYEc8yoNHaToMGBPaO+NlNc
KbqY1uRs+BYfK8QbjmDyjfzxI2MebDQK6tt5AOLOJ/DIGydsRCP35bPxXuDhqO+7
E3u5HMHGL0l5Sq0Uh4KOHRNM1NceXY5bopNZ
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
MIIEpjCCA46gAwIBAgIBADANBgkqhkiG9w0BAQQFADCBmDELMAkGA1UEBhMCVVMx
CzAJBgNVBAgTAkZMMREwDwYDVQQHEwhIZWF0aHJvdzEPMA0GA1UEChMGVHJpemVu
MRwwGgYDVQQLExNTb2Z3YXJlIEVuZ2luZWVyaW5nMRcwFQYDVQQDEw53d3cudHJp
emVuLmNvbTEhMB8GCSqGSIb3DQEJARYSemFyZW1iYUB0cml6ZW4uY29tMB4XDTAx
MTAyMDE1NDkxM1oXDTExMTAxODE1NDkxM1owgZgxCzAJBgNVBAYTAlVTMQswCQYD
VQQIEwJGTDERMA8GA1UEBxMISGVhdGhyb3cxDzANBgNVBAoTBlRyaXplbjEcMBoG
A1UECxMTU29md2FyZSBFbmdpbmVlcmluZzEXMBUGA1UEAxMOd3d3LnRyaXplbi5j
b20xITAfBgkqhkiG9w0BCQEWEnphcmVtYmFAdHJpemVuLmNvbTCCASIwDQYJKoZI
hvcNAQEBBQADggEPADCCAQoCggEBALPCypLQBuw53sRHHKuNK557iu6tibDcIKZZ
mYCal7Sww8Jd0gdLYGAFxJ5l0PwJZVxILUbbAiUCw2jQFPjq856HaxZ6s6smwWTZ
7piILEmbE46eKoTLkUybL3pwg43DuU6Zoizon+zNT1jwSoFGs1MMs3JdYhkuoGe8
kCF2KxRMxyY+o+RmdjSx158zG0S5yssSn2bbOjCeZzccRzQienVvZYbVahW3wrA8
j/oQubRhROD1jyDJRuDnZ+DvhhLxP0bdM11eUg9QQtl/ftYttL+RcgdORDB1GGV5
ELu79ukA/2UFh7QoZaBXZdje+uo2kITb97FKtjSMOrDOn/35GQECAwEAAaOB+DCB
9TAdBgNVHQ4EFgQUc2W51NgoxeGY7JOHJPQ6ZtoeY/swgcUGA1UdIwSBvTCBuoAU
c2W51NgoxeGY7JOHJPQ6ZtoeY/uhgZ6kgZswgZgxCzAJBgNVBAYTAlVTMQswCQYD
VQQIEwJGTDERMA8GA1UEBxMISGVhdGhyb3cxDzANBgNVBAoTBlRyaXplbjEcMBoG
A1UECxMTU29md2FyZSBFbmdpbmVlcmluZzEXMBUGA1UEAxMOd3d3LnRyaXplbi5j
b20xITAfBgkqhkiG9w0BCQEWEnphcmVtYmFAdHJpemVuLmNvbYIBADAMBgNVHRME
BTADAQH/MA0GCSqGSIb3DQEBBAUAA4IBAQAX41RvA0Q+j1uqo2o5Tym4/BO6NSR2
zaeAoOOnT8Sd+QHLNXpdC/Wz53tFgp1Kvkx9hwi+/wF082arpYMFaC63vSExl9v/
JMTiTiLcc0yC8z4XXVmgTr4CDRXurpTb4Zk6byeTxOn2pxXWFeOsEB/qIFHq+WDe
6QXizsMZ2YO60+cP7QAfsEnW5HIf8NOS6+KM6gbkWSgYpkZz5BeSidIDOilRSaEF
R4Yuox83E5RzcJ6xN0JzSD5AxvkRenhMLg7p59n/o8/JoASRg6ihovhs+UEXj6Z2
```

```
v+DdhwRuE/nLv5qt52jwUUj+zq1Wcinryb4QIErNaoPqlr2SyCkMlA4U
-----END CERTIFICATE-----
```

At this point you can go to the MS-DOS prompt where you stopped the s_client.exe application and press the up-arrow until you see the previous command. If this does not work just type in the previous command and press enter. You should see the following:



```
MRwwGgYDVQQLExNTb2Z3YXJlIEVuZ2luZWVyaW5nMRcwFQYDVQQDEw53d3cudHJp
emVuLmNvbTEhMB8GCSqGSIb3DQEJARYSemFyZW1iYUB0cml6ZW4uY29tMB4XDTAx
MTAyMDE4MTk0M1oXDTAyMTAyMDE4MTk0M1owgYwxCzAJBgNVBAYTAlVTMQswCQYD
VQQIEwJGTDERMA8GA1UEBxMITG9uZ3dvb2QxDDAKBgNVBAoTA1RTSTEMMAoGA1UE
CxMDRU5HMRwwGgYDVQQDExN3d3d3LnZpc3VhbDMyNzAuY29tMSMwIQYJKoZIhvcN
AQkBFhRyZWxhdGlvbnNAdHJpemVuLmNvbTCCASIwDQYJKoZIhvcNAQEBBQADggEP
ADCCAQoCggEBAMduDyCEUL3XYjgECElIg4NU3ISMdOAT5l9dzgiZQ+qdkshd3x27
3WeCq+esdeBs3dYYOEPfpxkBuEIEtu4cmJmB4CV+LJVot47w5kGJitFC3yjDLtqv
5q/PcT/vKf5jb2DA+iw7KlOwwO7tXCTKBSStP2esMcTRWP20iuwwt5hDAGrG1h8v
UZMQdxM93nFAhf5RkrYc1plF770M6oeU1OFS8RuMGUjpOVWbSvoUOCzMTh95FV6Z
A1p9SA9AdPGLz3TuBcDAftXGi9kUY7vZis4yIKUh5U6VmrKSfpNhsmgeDu8l8XVt
eiQqAG0B/oFQlv0DH3WdAAOHKe02I71FiysCAwEAAaOCASQwggEgMAkGA1UdEwQC
MAAwLAYJYIZIAYb4QgENBB8WHU9wZW5TU0wgR2VuZXJhdGVkIENlcnRpZmljYXRl
MB0GA1UdDgQWBBRlqnayQK4E8kcupDrR6D8wiyssMzCBxQYDVR0jBIG9MIG6gBRz
ZbnU2CjF4Zjsk4ck9Dpm2h5j+6GBnqSBmzCBmDELMAkGA1UEBhMCVVMxCzAJBgNV
BAgTAkZMMREwDwYDVQQHEwhIZWF0aHJvdzEPMA0GA1UEChMGVHJpemVuMRwwGgYD
VQQLExNTb2Z3YXJlIEVuZ2luZWVyaW5nMRcwFQYDVQQDEw53d3cudHJpemVuLmNv
bTEhMB8GCSqGSIb3DQEJARYSemFyZW1iYUB0cml6ZW4uY29tggEAMA0GCSqGSIb3
DQEBBAUAA4IBAQAh+/rh0AUEhogw6ejX2+N/0vxX//bBoRIfqsTydp7ImI9lrszl
cx8yXMSX+jP6sdt34taGJVg7uJAaYL+QVXiOV97OcsEY196rYToc9QaZM6iYEQYd
7KxohGtx59s9ehE0a+R1mwqVB/7QvE/pjXXfazYNw9WZyxTcJ8HzgsLDJYPg0kx5
Zt4/6P4iu4nVXZdYoqiZHQBJB1X5wcf8eDjHkgJdWRYEc8yoNHaToMGBPaO+NlNc
KbqY1uRs+BYfK8QbjmDyjfzxI2MebDQK6tt5AOLOJ/DIGydsRCP35bPxXuDhqO+7
E3u5HMHGL0l5Sq0Uh4KOHRNM1NceXY5bopNZ
-----END CERTIFICATE-----
subject=/C=US/ST=FL/L=Longwood/O=TSI/OU=ENG/CN=wwww.visual3270.com/Email=relatio
ns@trizen.com
issuer=/C=US/ST=FL/L=Heathrow/O=Trizen/OU=Sofware Engineering/CN=www.trizen.com/
Email=zaremba@trizen.com
---
No client certificate CA names sent
---
SSL handshake has read 1783 bytes and written 256 bytes
---
New, TLSv1/SSLv3, Cipher is EDH-RSA-DES-CBC3-SHA
Server public key is 2048 bit
SSL-Session:
    Protocol  : TLSv1
    Cipher    : EDH-RSA-DES-CBC3-SHA
    Session-ID: 9FAC4D9C7B89915A3B18F11A6A1B627864 40C6ED61D4359EA4BD45BBB4155C9F

    Session-ID-ctx:
    Master-Key: 66266E78ABD162FA116BD2FBF8EE0EE406256799BD25C5C61E59A3525E205105
4C991859DA0D4B0870C86BB2933BE546
    Key-Arg   : None
    Start Time: 1003624026
    Timeout   : 300 (sec)
    Verify return code: 0 (ok)
---
```

Notice "*Verify return code: 0 (ok)"* response for the client verification! This is what we wanted from the beginning. So did we need the cert.crt certificate in the root.pem? The answer is no, as long as you have the top level certificate installed in the CAFile (root.pem) the client will return "ok."

OpenSSL Utilities

S_SERVER

s_server is a great tool for testing your SSL client applications; however, it only handles one request at a time, so if you're interested in testing your applications against an SSL server then purchase Visual SSL for ActiveX, Delphi, or C++Builder which has a multi-threaded blocking server and an Asynchronous non-blocking server and source code to match. The following selected "switches" are available for use within s_server. An example follows each switch.

-accept [port]

The TCP port the server should "listen" on. The default is 4433. The next line will start a server listening on port 3000.
s_server –accept 3000

-cert [certname]

This is the public certificate to use and can be a physical path to the certificate. Note that if you use a DSS cipher suite then you must use a DSS certificate as well as the matching private key file (see GENDSA). If a certificate is not specified then the server will attempt to load server.pem in the local directory. If the public and private key do not match the server will exit. Developers should note that if a DSS certificate is specified then only DSS clients will be accepted and is the same for RSA certificates and RSA clients.
s_server –accept 3000 –cert c:\CARoot\private\ca.crt

-key [keyfile]

This is the private key file to use and can be a physical path to the certificate. If no key file is presented then the server will attempt to load the key from the certificate file (key file data and certificate data can exist in the same file). If no key data can be found the server will attempt to load "server.pem." Developers should provide their own public and key files as described above. Servers must have a public and private key file and it is preferred to encrypt the key file as described earlier. Notice you can use complete paths or relative paths.
s_server –key c:\CARoot\private\ca.key –cert c:\CARoot\private\ca.crt

-no_tmp_rsa

Some export (EXP) cipher suites generate a temporary RSA key. By setting this option no temporary RSA key will be created. Consequently, a client that only supports EXP type cipher suites may fail the handshake and not be able to connect. Developers, when using client connections should try to connect first with a US domestic cipher.
s_server –no_tmp_rsa –accept 3000 –cert ca.crt –key ca.key

-verify [depth], -Verify [depth]

As discussed earlier, a client's certificate can be requested by the server for verification. By using the **–verify** [value] the client's certificate is requested with a certificate chain depth specified by [value] but will not shutdown if the client doesn't present a certificate. The **–Verify** [depth] switch is the same as the lowercase version but will shutdown the connection if the certificate is not presented for verification. As mentioned earlier, the certificate chain represents the number of Certificate Authorities that have "signed" the certificate and then the actual issued certificate. By setting the [depth] integer value, developers can set the maximum number of CA's that are in the chain including the issued certificate. Most certificate chains will have 1 or 2 CA's.

s_server –Verify 1 –cert ca.pem –key key.pem –accept 2000

-CApath [directory]

This switch represents a directory where the acceptable certificates are kept, usually within a CA directory structure. The important thing to note is this directory must have the **perl** script utility **c_rehash** (in tools directory of OpenSSL) run regularly on the certificates in the directory. It is easier for Windows developers to utilize the **–CAFile** [file] switch.

s_server –Verify 1 –CAPath c:\CARoot\certs

-CAfile [file]

This switch points to the file that contains a list of all acceptable certificates and acceptable Certificate Authorities. See earlier discussion on client verification on how to create this file.

s_server –CAFile c:\CARoot\root.pem

-state

This option will display each SSL protocol step in the handshake upon the acceptance of a client connection.

s_server –state –accept 4433 –cert ca.crt –key ca.key

-debug

selecting this option/switch will produce an exorbitant amount of information on the client connection including a hex dump. This option is best if analyzing the entire protocol from start to finish and is best used with NT's cmd.exe DOS prompt.

s_server –debug –accept 4433 –no_tmp_rsa

-nbio

This switch turns on non-blocking connections which don't block execution until a read/write is performed. Non-blocking connections are usually associated with asynchronous event notification and are used primarily for non serialized connections such as a chat program. Non-blocking connections continuously call Accept until the client connection is accepted.

s_server –accept 4000 –nbio

-quiet

This is basically the opposite of any “debug” switches turning all reporting of the client connections off.
s_server –accept 3000 –quiet

-ssl2, -ssl3, -tls1, -no_ssl2, -no_ssl3, -no_tls1

These switches disable or enable the use of certain SSL or TLS protocols. For example, the –ssl3 switch will force only SSL version 3 connections. SSL version 3 is probably the most widespread with respect to new technologies and should be used most of the time. The TLSv1 is basically the same as SSL version 3 and is gaining acceptance rapidly.
s_server –accept 3000 –ssl3

-bugs

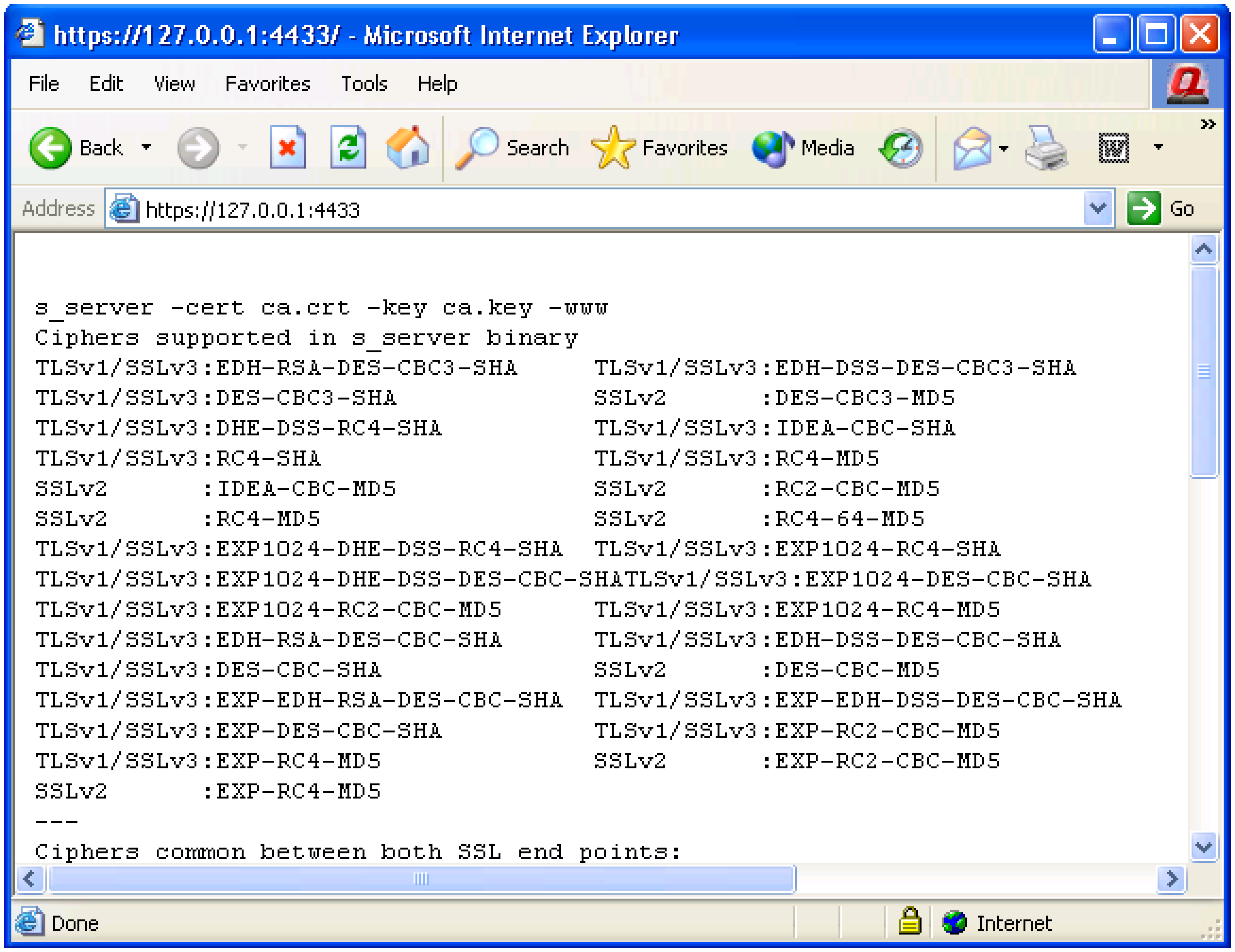
This switch will enable workarounds for known SSL or TLS bugs.
s_server –bugs

-cipher [cipherlist]

This option will set the server’s acceptable ciphers. If not specified, the server will default to accepting any of the available ciphers in the OpenSSL library; however, you may want to test a server or client’s ability or inability to find a “shared cipher.” If utilized, upon the client’s initial connection the first matching pair between the client and server (using the client’s list) will be selected. If the server does not support any of the client’s preferred ciphers, the connection will be closed with “no shared ciphers.” Acceptable ciphers should be “colon” delimited.
s_server –accept 443 –cipher EXP-RC4-MD5:RC4-MD5

-www

Use this option to check a web browser’s connection to the server. The server will return the connection information back to the web browser in HTML format. After starting the server using an RSA certificate, developers can enter https://127.0.0.1:4433 into the browser to see the connection information returned from the server assuming the server and the web browser are on the same machine.
s_server –accept 4433 –www


https://127.0.0.1:4433/ - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Favorites
Media
Address https://127.0.0.1:4433
Go
Done
Internet

```
s_server -cert ca.crt -key ca.key -www
Ciphers supported in s_server binary
TLSv1/SSLv3:EDH-RSA-DES-CBC3-SHA      TLSv1/SSLv3:EDH-DSS-DES-CBC3-SHA
TLSv1/SSLv3:DES-CBC3-SHA              SSLv2      :DES-CBC3-MD5
TLSv1/SSLv3:DHE-DSS-RC4-SHA           TLSv1/SSLv3:IDEA-CBC-SHA
TLSv1/SSLv3:RC4-SHA                   TLSv1/SSLv3:RC4-MD5
SSLv2      :IDEA-CBC-MD5              SSLv2      :RC2-CBC-MD5
SSLv2      :RC4-MD5                   SSLv2      :RC4-64-MD5
TLSv1/SSLv3:EXP1024-DHE-DSS-RC4-SHA  TLSv1/SSLv3:EXP1024-RC4-SHA
TLSv1/SSLv3:EXP1024-DHE-DSS-DES-CBC-SHATLSv1/SSLv3:EXP1024-DES-CBC-SHA
TLSv1/SSLv3:EXP1024-RC2-CBC-MD5       TLSv1/SSLv3:EXP1024-RC4-MD5
TLSv1/SSLv3:EDH-RSA-DES-CBC-SHA       TLSv1/SSLv3:EDH-DSS-DES-CBC-SHA
TLSv1/SSLv3:DES-CBC-SHA               SSLv2      :DES-CBC-MD5
TLSv1/SSLv3:EXP-EDH-RSA-DES-CBC-SHA  TLSv1/SSLv3:EXP-EDH-DSS-DES-CBC-SHA
TLSv1/SSLv3:EXP-DES-CBC-SHA           TLSv1/SSLv3:EXP-RC2-CBC-MD5
TLSv1/SSLv3:EXP-RC4-MD5               SSLv2      :EXP-RC2-CBC-MD5
SSLv2      :EXP-RC4-MD5
---
Ciphers common between both SSL end points:
```

-WWW

This switch will emulate a web server and return the request page if it exists in the server's executable directory. For instance if http://127.0.0.1:4433/page.html is requested, then page.html will be returned if it exists in the server's executable directory. Developers should make sure that the file first exists before making this test.

-rand [file(s)]

This switch specifies the location of a file that has random data within it or multiple files that can be used to seed the random number generator. If multiple files are specified then Windows users must use a ";" semicolon delimiter. It has been reported that hacking can be achieved by generating the same random number and it is recommended to use something such as a bit map file.
s_server –accept 443 –cert cert.pem –key key.pem –rand .rnd

s_server runtime commands

These commands can be entered into your server during its execution:

Key	Description
Q	By entering a single 'q' this will end the current SSL connection.
Q	Enter this to close the current connection and exit the server.
R	This will renegotiate the SSL client connection.
R	This will renegotiate the SSL client connection and request the client's certificate.
P	send some plain text down the underlying TCP connection: this should cause the client to disconnect due to a protocol violation.
S	print out some session cache status information

S_CLIENT

s_client is a great utility for testing your SSL servers and has a number of options that will make debugging your applications a little easier. The following selected switches will help you make your applications SSL compliant:

-connect [host:port]

This switch is used to connect to the specified host. If no host is specified then the client will default to 127.0.0.1 port 4433.

s_client –connect 127.0.0.1:4433 –cert ca.crt –key ca.key

-cert [cert file]

This will specify the public certificate to use during an SSL handshake. This is only needed if the server requests a client's certificate. It should be noted that most http servers do not request a certificate and as such the default for s_client is no certificate.

s_client –connect 127.0.0.1:4433 –cert cert.pem –key cert.key

-key [keyfile]

Like the s_server –key switch, this switch will load the private key file. If no key file is used the client application will attempt to use the public certificate. If the public certificate does not contain any private key information then client will not use a certificate.

s_client –connect 127.0.0.1:4433 –cert ca.crt –key ca.key

-verify [depth]

This specified the server's certificate chain depth to use and turns on the server certificate verification process. The [depth] specifies the maximum number of CA's that can be present in the server's certificate chain, which will usually be 1 or 2. Even if a server's certificate fails the verification process, the connection is still made. In order to verify the certificate completely developers must also use the CAFile switch to load the CA's that are trusted and the associated certificates

to trust in order to get a complete Ok from the verification process as discussed earlier.
s_client –connect 127.0.0.1:3000 –verify 2 –cert c:\CARoot\cert.pem

-CApath [directory]
This switch represents a directory where the acceptable server certificates are kept, usually within a CA directory structure. The important thing to note is this directory must have the **perl** script utility **c_rehash** (in tools directory of OpenSSL) run regularly on the certificates in the directory. It is easier for Windows developers to utilize the **–CAFile** [file] switch.
s_client –verify 2 –CAPath c:\CARoot\certs.

-CAfile [file]
This switch points to the file that contains a list of all the acceptable server certificates and acceptable Certificate Authorities. See earlier discussion on client verification on how to create this file.
s_client –CAFile c:\CARoot\root.pem

-reconnect
This switch will connect to the server 5 consecutive times and is useful for minor stress testing and session cache testing.
s_client –reconnect –connect 127.0.0.1:443

-showcerts
This will display the complete server certificate chain. Normal connection will only show the server's certificate information.
s_client –connect 127.0.0.1:3000 –showcerts

-state
This option will display each SSL protocol step in the handshake upon the acceptance of a client connection.
s_client –state –connect 127.0.0.1:4433 –cert ca.crt –key ca.key

-debug
Selecting this option/switch will produce an exorbitant amount of information on the server connection including a hex dump. This option is best if analyzing the entire protocol from start to finish and is best used with NT's cmd.exe DOS prompt.
s_client –debug –connect 127.0.0.1:4433

-nbio
This switch turns on the non-blocking connection type, which doesn't block execution while waiting for a read/write. Non blocking connections are usually associated with asynchronous event notification and are used primarily for non serialized connections such as a chat program. Non-blocking connections continuously call Accept until the client connection is accepted.

s_client –connect 127.0.0.1:4000 –nbio

-quiet

This is basically the opposite of any “debug” switches turning all reporting of the client connections off.
s_client –connect 127.0.0.1:3000 –quiet

-ssl2, -ssl3, -tls1, -no_ssl2, -no_ssl3, -no_tls1

These switches disable or enable the use of certain SSL or TLS protocols. For example, the –ssl3 switch will force only SSL version 3 connections. SSL version 3 is probably the most widespread with respect to new technologies and should be used most of the time. The TLSv1 is basically the same as SSL version 3 and is gaining acceptance rapidly.
s_client –accept 3000 –ssl3

-bugs

This switch will enable workarounds for known SSL or TLS bugs.
s_client –bugs

-cipher [cipherlist]

This option will set the client’s preferred ciphers in order of preference. If not specified, the client will default to presenting any of the available ciphers in the OpenSSL library; however, you may want to test a server or client’s ability or inability to find a “shared cipher.” If utilized, upon the client’s initial connection the first matching pair between the client and server (using the client’s list) will be selected. If the server does not support any of the client’s preferred ciphers, the connection will be closed with “no shared ciphers.” Acceptable ciphers should be “colon” delimited.
s_client –connect 127.0.0.1:443 –cipher EXP-RC4-MD5:RC4-MD5

-rand [file(s)]

This switch specifies the location of a file that has random data within it or multiple files that can be used to seed the random number generator. If multiple files are specified then Windows users must use a “;” semicolon delimiter. It has been reported that hacking an encrypted session can be achieved by generating the same random number and it is recommended to use something such as a bit map file.
s_client –connect 127.0.0.1:443 –cert cert.pem –key key.pem –rand .rnd

ASN1PARSE

The ASN1Parse utility will take either a PEM (Privacy Enhanced Mail RFC1422) or DER (Distinguished Encoding Rules) public certificate and parse the certificate information, displaying the information. The PEM format consists of the DER base64 encoding while the standard DER format is compatible with RFC2459; both of which are based on the ASN.1 (Abstract Syntax Notation 1) and include X.509 information as well. There are several options that can be used but the most useful are as follows:

-in [certificate]
This options specifies the certificate file. The default certificate format is PEM and as such the –inform and –in file must match.

-inform [PEM | DER]
This option sets the type of file that will be parsed.

asn1parse –in ca.pem –inform PEM

DSAPARAM

This utility is useful for generating DSA parameter files. DSA, as opposed to RSA, utilizes the Digital Signature Algorithm now known as the Digital Signature Standard (DSS) and can only use SHA-1 whereas RSA can utilize SHA-1 and MD5 digest algorithms. It is really only important to know that a DSA/DSS client cannot communicate with an RSA server. There is really only one reason to use this utility and it is to create the DSA parameter file, which can then be used to create a DSA key pair.

-out [file]
This specifies the out file that will have the DSA parameters. The out file will default to a PEM format unless DER is specified in the –outform.

-outform [PEM | DER]
This switch specifies the type of out file to create.

[number]
This is not a switch but a parameter and represents the number of bits to use when generating the parameters. This must be the last parameter in the command.

-rand [file(s)]
This switch specifies the location of a file that has random data within it or multiple files that can be used to seed the random number generator. If multiple files are specified then Windows users must use a ";" semicolon delimiter. It has been reported that hacking an encrypted session can be achieved by generating the

same random number and it is recommended to use something such as a bit map file or a privately used file that has random data.

dsaparam –outform PEM –out dsaparams.pem –rand .rnd 512

GENDSA

This utility will generate a DSA (Digital Signature Algorithm) key from a DH parameters file that has been created using the DSAPARAM utility.

-out [file]
This specifies the out file name of the DSA generated private key file.

-des, -des3, -idea
This switches/options represent the encryption format for the key. To encrypt the key select one of the above algorithms or leave the key file unencrypted. This will prompt the user for a password.

-rand [file(s)]
This switch specifies the location of a file that has random data within it or multiple files that can be used to seed the random number generator. If multiple files are specified then Windows users must use a ";" semicolon delimiter. It has been reported that hacking an encrypted session can be achieved by generating the same random number and it is recommended to use something such as a bit map file or a privately used file that has random data.

dsaparam-file
A command line parameter rather than a switch, this selects the DSA parameters file to use when generating the DSA key. The DSA parameters file be created using the DSAPARAM utility.

gendsa –out dsa.key –des3 –rand c:\private\.rnd dsaparam.pem

RAND

This utility will generate a random byte file for use with the –rand switch in most of the OpenSSL applications. Many times it is just as good to use a file such as a picture, or word document to seed the random number generator during the openssl commands; however, for simplicity it may be easier to always use the same random bytes file.

-out [file]
This sets the file to create which will have the random bytes. In the OpenSSL documentation and for Certificate Authority management, this file typically consists of only the extension .rnd.

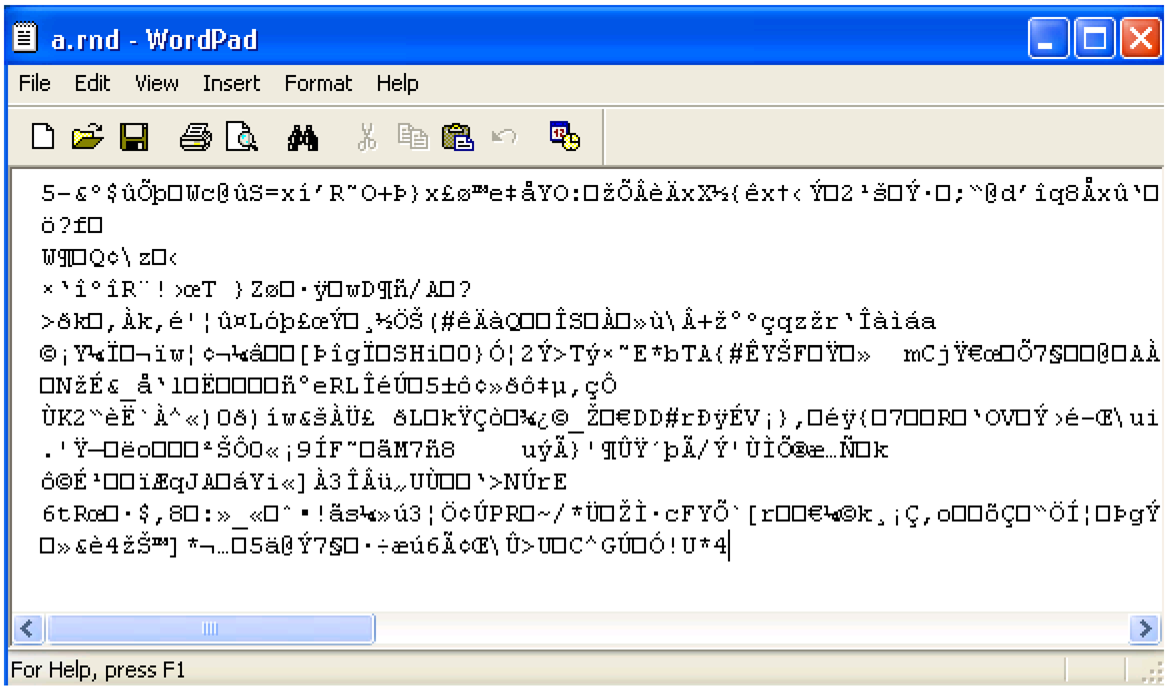
-rand [file(s)]

This switch specifies the location of a file that has random data within it or multiple files that can be used to seed the random number generator. If multiple files are specified then Windows users must use a ";" semicolon delimiter. It has been reported that hacking an encrypted session can be achieved by generating the same random number and it is recommended to use something such as a bit map file or a privately used file that has random data.

[num]

This parameter, rather than a switch, sets the number of random bytes to create and place into the file. It is recommended to have this file at least 1K or 1024 bytes.

C:\>*rand –out .rnd 1024*



```
5-&°$ûÕþ□Wc@ûS=xí'R˜O+Þ}x£ø™e‡åYO:□žÕÅèÄxX½{êxt<Ý□2¹š□Ý·□;˜@d'íq8Åxû'□
ö?f□
W¶□Q¢\z□<
×'ì°ìR¨!›œT }Zø□·ÿ□wD¶ñ/A□?
>ðk□,Àk,é'¦û¤Lóþ£œÝ□¸½ÖŠ(#êÄàQ□□ÎS□À□»ù\Â+ž°°çqzžr'Îàìáa
©¡Ý¼Ï□¬ìw¦¢¬¼â□□[ÞìgÏ□SHi□0)Ó¦2Ý>Tý×˜E*bTA{#ÊÝŠF□Ý□»  mCjŸ€œ□Õ7§□□@□AÀ
□NžÉ&_å'1□Ë□□□□ñ°eRLÎèÚ□5±ô¢»ðô‡µ,çÔ
ÙK2˜èË`À^«)O ð)ìw&šÀÜ£ ðL□kŸçò□¾¿©_Ž□€DD#rÐÿÉV¡},□éÿ{□7□□R□'OV□Ý›é–Œ\ui
.'Ÿ–□ëo□□□²ŠÓO«¡9ÍF˜□ãM7ñ8    uýÃ}'¶ÛŸ´þÃ/Ý'ÙÌÕ®æ…Ñ□k
ô©É¹□□iÆqJA□áYi«]À3ÎÅù„UÙ□□'>NÚrE
6tRœ□·$,8□:»_«□ˆ▪!ãs¼»ú3¦Ö¢ÚPR□~/*Ü□ŽÌ·cFYÕ`[r□□€¼©k¸¡Ç,o□□õÇ□˜ÖÍ¦□ÞgÝ
□»&è4žŠ™]*¬…□5ä@Ý7§□·÷æú6Ã¢Œ\Û>U□C^GÚ□Ó!U*4
```

Windows developers should note that usually the only way to open a .rnd file is to edit the file from within a DOS prompt by using the *edit .rnd* command.

VERIFY

This utility will verify the presented certificate similar to the certificate verification process that occurs during the SSL handshake where a client and server both verify each other if requested. Developers can use this utility to extract the code from the verification process as outlined below:

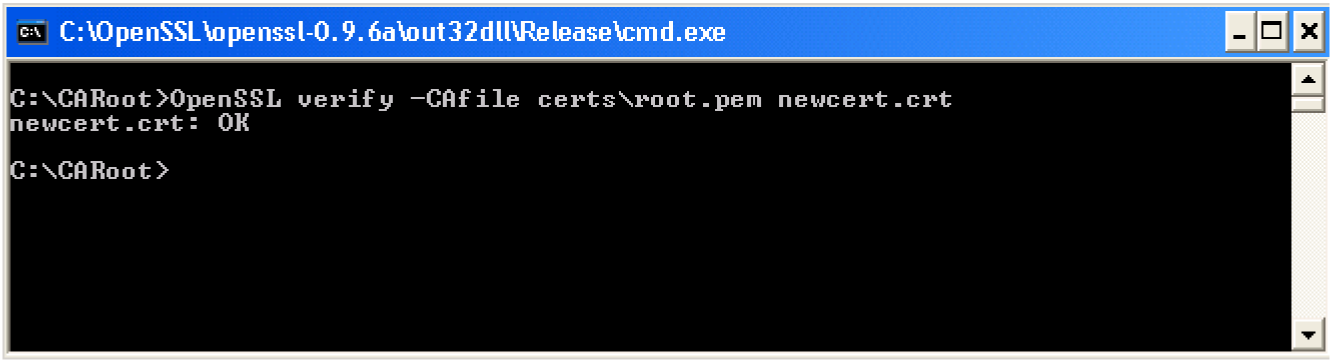
-CAFile [file]

This parameter is the CAFile to use when building the certificate chain. The chain built up by starting with the selected certificate and ending with the

CARoot. A certificate is self signed if the certificate and certificate root are the same. If a match between the certificate and the CAFile is found, then the certificate if verified ok.

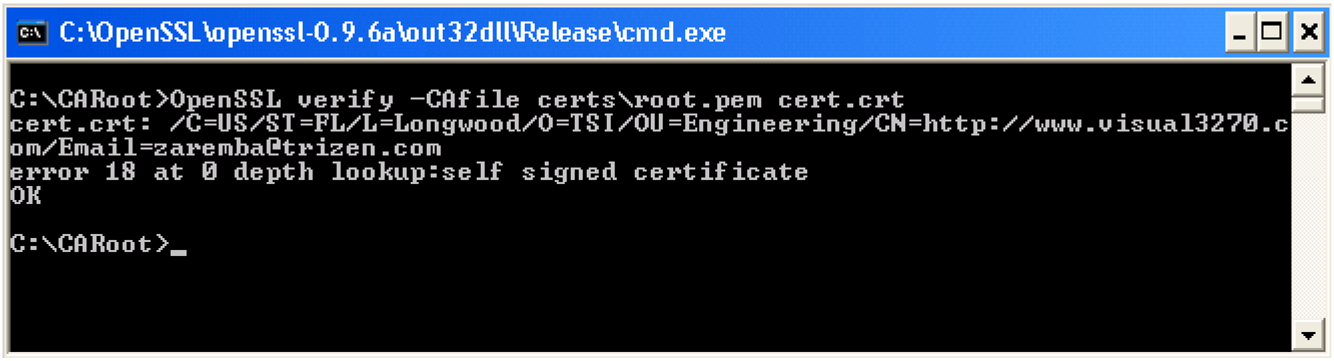
[certificate1 certificate2 .. certificateN]
This specifies the certificates to validate/verify.

C:\>OpenSSL verify –CAfile c:\CARoot\root.pem c:\CARoot\newcert.crt


C:\OpenSSL\openssl-0.9.6a\out32dll\Release\cmd.exe

```
C:\CARoot>OpenSSL verify -CAfile certs\root.pem newcert.crt
newcert.crt: OK

C:\CARoot>
```

Notice that the return value was OK. This means the certificate returned a verify result of zero (0). If you have followed the examples for creating a certificate and adding your CA root certificate to the root.pem file specified above then you should have similar results. Another possible result could be the following:


C:\OpenSSL\openssl-0.9.6a\out32dll\Release\cmd.exe

```
C:\CARoot>OpenSSL verify -CAfile certs\root.pem cert.crt
cert.crt: /C=US/ST=FL/L=Longwood/O=TSI/OU=Engineering/CN=http://www.visual3270.c
om/Email=zaremba@trizen.com
error 18 at 0 depth lookup:self signed certificate
OK

C:\CARoot>_
```

In this example, we have selected to verify a certificate that was generated from a key whose public certificate is NOT in the root.pem file and has not been signed. This error is typical of “dummy” certificates, created but not signed. Other errors may be a certificate that has been signed, but the issuer is not in the root.pem CAfile specification.

The following represents the possible errors, defined by the OpenSSL API, that can be received during the verification process and in fact can be seen in the Visual SSL event OnVerifyError:

Code	Defined	Long Description
0	X509_V_OK	The operation was successful.
2	X509_V_ERR_UNABLE_TO_GET_ISSUER_CERT	The issuer certificate could not be found: this occurs if the issuer certificate of an untrusted certificate cannot be found.
3	X509_V_ERR_UNABLE_TO_GET_CRL	The CRL of a certificate could not be found. Unused.
4	X509_V_ERR_UNABLE_TO_DECRYPT_CERT_SIGNATURE	the certificate signature could not be decrypted. This means that the actual signature value could not be determined rather than it not matching the expected value, this is only meaningful for RSA keys.
5	X509_V_ERR_UNABLE_TO_DECRYPT_CRL_SIGNATURE	The CRL signature could not be decrypted: this means that the actual signature value could not be determined rather than it not matching the expected value. Unused.
6	X509_V_ERR_UNABLE_TO_DECODE_ISSUER_PUBLIC_KEY	The public key in the certificate SubjectPublicKeyInfo could not be read.
7	X509_V_ERR_CERT_SIGNATURE_FAILURE	The signature of the certificate is invalid.
8	X509_V_ERR_CRL_SIGNATURE_FAILURE	The signature of the certificate is invalid. Unused.
9	X509_V_ERR_CERT_NOT_YET_VALID	The certificate is not yet valid: the notBefore date is after the current time.
10	X509_V_ERR_CRL_NOT_YET_VALID	The CRL is not yet valid. Unused.
11	X509_V_ERR_CERT_HAS_EXPIRED	The certificate has expired: that is the notAfter date is before the current time.
12	X509_V_ERR_CRL_HAS_EXPIRED	The CRL has expired. Unused.
13	X509_V_ERR_ERROR_IN_CERT_NOT_BEFORE_FIELD	the certificate notBefore field contains an invalid time.
14	X509_V_ERR_ERROR_IN_CERT_NOT_AFTER_FIELD	the certificate notAfter field contains an invalid time.
15	X509_V_ERR_ERROR_IN_CRL_LAST_UPDATE_FIELD	the CRL lastUpdate field contains an invalid time. Unused.
16	X509_V_ERR_ERROR_IN_CRL_NEXT_UPDATE_FIELD	the CRL nextUpdate field contains an invalid time. Unused.
17	X509_V_ERR_OUT_OF_MEM	an error occurred trying to allocate memory. This should never happen.
18	X509_V_ERR_DEPTH_ZERO_SELF_SIGNED_CERT	the passed certificate is self signed and the same certificate cannot be found in the list of trusted certificates.
19	X509_V_ERR_SELF_SIGNED_CERT_IN_CHAIN	the certificate chain could be

		built up using the untrusted certificates but the root could not be found locally.
20	X509_V_ERR_UNABLE_TO_GET_ISSUER_CERT_LOCALLY	the issuer certificate of a locally looked up certificate could not be found. This normally means the list of trusted certificates is not complete.
21	X509_V_ERR_UNABLE_TO_VERIFY_LEAF_SIGNATURE	no signatures could be verified because the chain contains only one certificate and it is not self signed.
22	X509_V_ERR_CERT_CHAIN_TOO_LONG	the certificate chain length is greater than the supplied maximum depth. Unused.
23	X509_V_ERR_CERT_REVOKED	the certificate has been revoked. Unused.
24	X509_V_ERR_INVALID_CA	a CA certificate is invalid. Either it is not a CA or its extensions are not consistent with the supplied purpose.
25	X509_V_ERR_PATH_LENGTH_EXCEEDED	the basicConstraints pathlength parameter has been exceeded.
26	X509_V_ERR_INVALID_PURPOSE	the supplied certificate cannot be used for the specified purpose.
27	X509_V_ERR_CERT_UNTRUSTED	the root CA is not marked as trusted for the specified purpose.
28	X509_V_ERR_CERT_REJECTED	the root CA is marked to reject the specified purpose.
29	X509_V_ERR_SUBJECT_ISSUER_MISMATCH	the current candidate issuer certificate was rejected because its subject name did not match the issuer name of the current certificate. Only displayed when the *-issuer_checks* option is set.
30	X509_V_ERR_AKID_SKID_MISMATCH	the current candidate issuer certificate was rejected because its subject key identifier was present and did not match the authority key identifier current certificate. Only displayed when the *-issuer_checks* option is set.
31	X509_V_ERR_AKID_ISSUER_SERIAL_MISMATCH	the current candidate issuer certificate was rejected because its issuer name and serial number was present and did not match the authority key identifier of the current certificate. Only displayed when the *-issuer_checks* option is set.

32	`X509_V_ERR_KEYUSAGE_NO_CERTSIGN`	`the current candidate issuer certificate was rejected because its keyUsage extension does not permit certificate signing.`
50	`X509_V_ERR_APPLICATION_VERIFICATION`	`an application specific error. Unused.`

enc

The ENC utility will encrypt and decrypt files using the selected algorithm. The format is typically the following:

C:\>OpenSSL [ciphername] [options]
Or
C:\>OpenSSL enc [ciphername] [options]

-in [filename]
This option sets the file to decrypt or encrypt.

-out [filename]
This option sets the output file.

-salt
This option will add a salt (random) value to the encryption. It is advised to use this option when encrypting files.

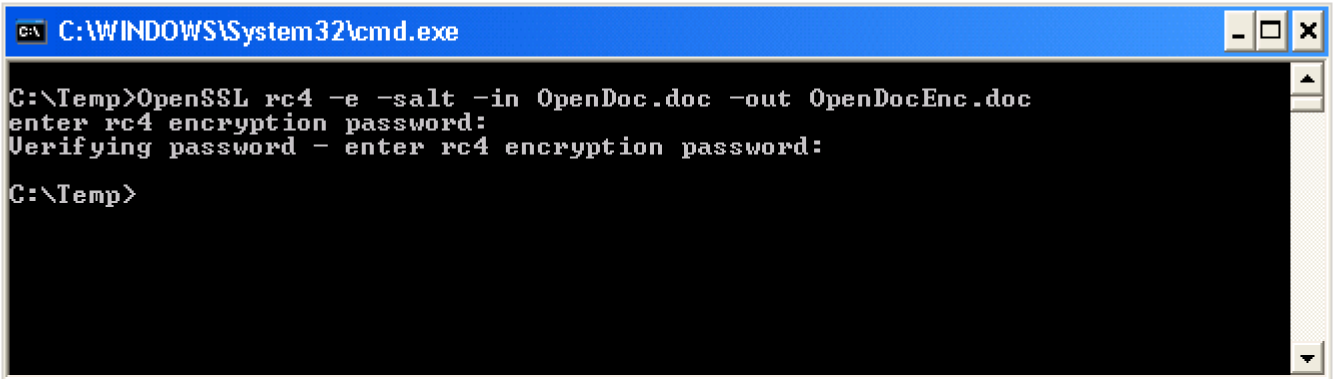
-e
Encrypt the specified data file. This is the default.

-d
Decrypt the specified data file

The following list represents the supported ciphers that can be used when encrypting or decrypting data.

Cipher	Description
base64	Base 64
bf-cbc	Blowfish in CBC mode
Bf	Alias for bf-cbc
bf-cfb	Blowfish in CFB mode
bf-ecb	Blowfish in ECB mode
bf-ofb	Blowfish in OFB mode
cast-cbc	CAST in CBC mode
Cast	Alias for cast-cbc
cast5-cbc	CAST5 in CBC mode
cast5-cfb	CAST5 in CFB mode
cast5-ecb	CAST5 in ECB mode
cast5-ofb	CAST5 in OFB mode

des-cbc	DES in CBC mode
des	Alias for des-cbc
des-cfb	DES in CBC mode
des-ofb	DES in OFB mode
des-ecb	DES in ECB mode
des-ede-cbc	Two key triple DES EDE in CBC mode
des-ede	Alias for des-ede
des-ede-cfb	Two key triple DES EDE in CFB mode
des-ede-ofb	Two key triple DES EDE in OFB mode
des-ede3-cbc	Three key triple DES EDE in CBC mode
des-ede3	Alias for des-ede3-cbc
des3	Alias for des-ede3-cbc
des-ede3-cfb	Three key triple DES EDE CFB mode
des-ede3-ofb	Three key triple DES EDE in OFB mode
desx	DESX algorithm.
idea-cbc	IDEA algorithm in CBC mode
idea	same as idea-cbc
idea-cfb	IDEA in CFB mode
idea-ecb	IDEA in ECB mode
idea-ofb	IDEA in OFB mode
rc2-cbc	128 bit RC2 in CBC mode
rc2	Alias for rc2-cbc
rc2-cfb	128 bit RC2 in CBC mode
rc2-ecb	128 bit RC2 in CBC mode
rc2-ofb	128 bit RC2 in CBC mode
rc2-64-cbc	64 bit RC2 in CBC mode
rc2-40-cbc	40 bit RC2 in CBC mode
rc4	128 bit RC4
rc4-64	64 bit RC4
rc4-40	40 bit RC4
rc5-cbc	RC5 cipher in CBC mode
rc5	Alias for rc5-cbc
rc5-cfb	RC5 cipher in CBC mode
rc5-ecb	RC5 cipher in CBC mode
rc5-ofb	RC5 cipher in CBC mode



```
C:\Temp>OpenSSL rc4 -e -salt -in OpenDoc.doc -out OpenDocEnc.doc
enter rc4 encryption password:
Verifying password - enter rc4 encryption password:

C:\Temp>
```

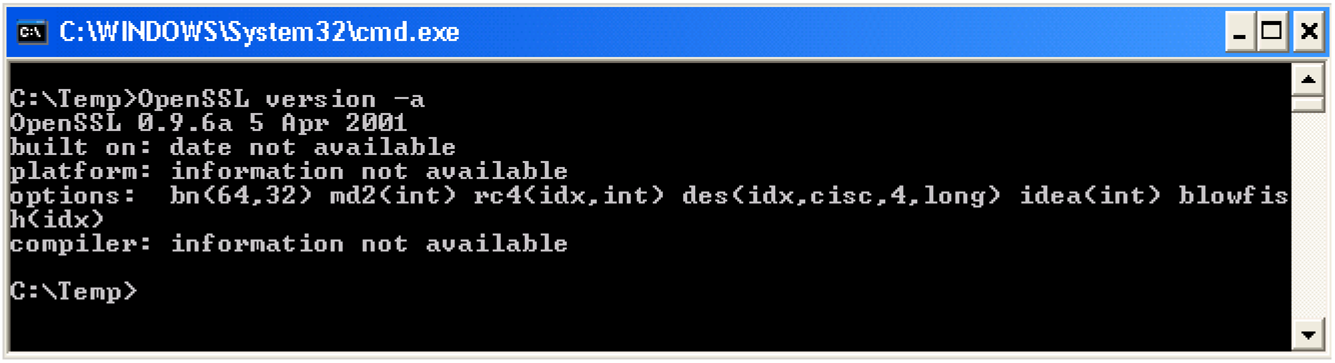
In the above example we have chosen to encrypt a Word document using the rc4 cipher with a salt. The size of the file is the document you are reading and is approximately 22MB. This encryption only takes a few seconds to complete. The next example shows the decryption of the same file.

C:\>OpenSSL rc4 –d –in OpenDocEnc.doc –out OpenDocDec.doc

version

This utility will return the OpenSSL version being utilized.

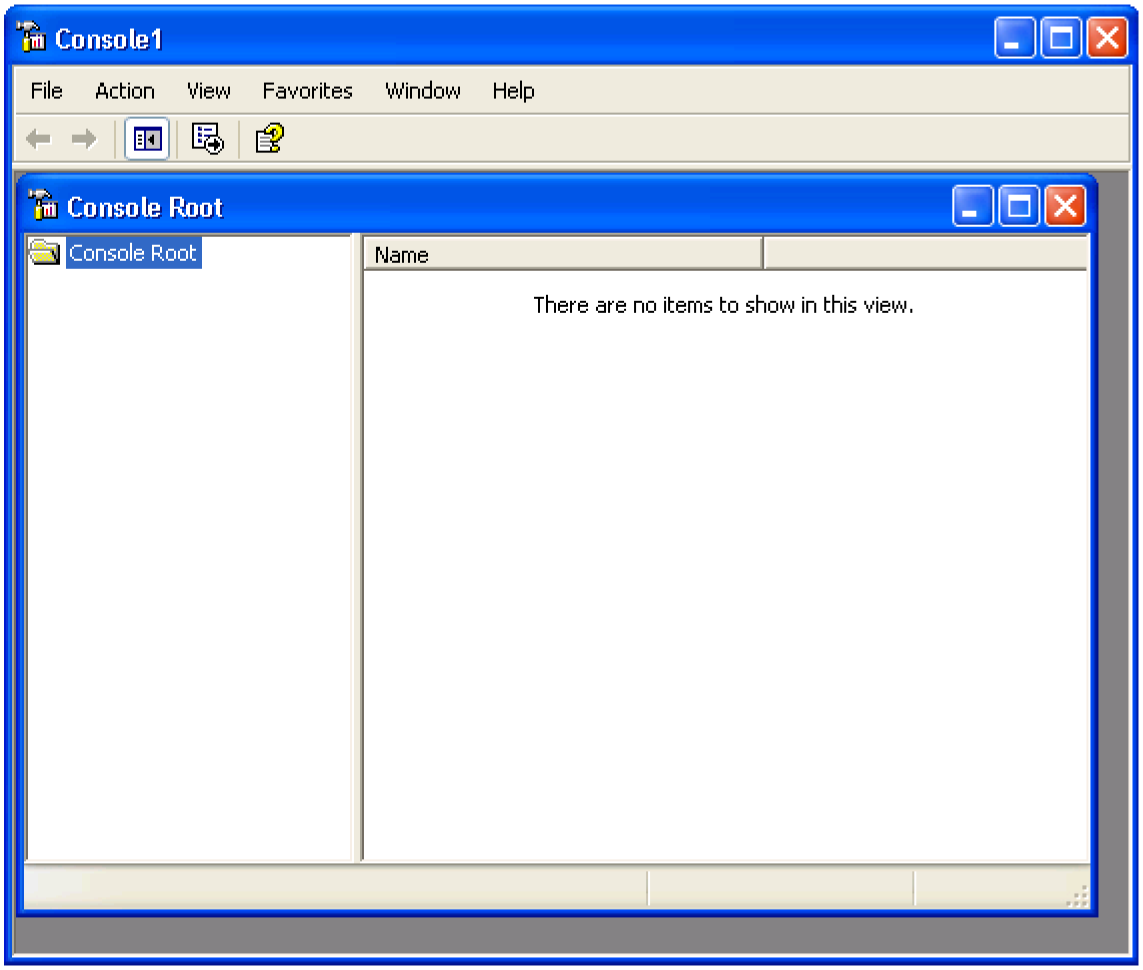
-a
All information concerning the OpenSSL library.



```
C:\Temp>OpenSSL version -a
OpenSSL 0.9.6a 5 Apr 2001
built on: date not available
platform: information not available
options:  bn(64,32) md2(int) rc4(idx,int) des(idx,cisc,4,long) idea(int) blowfis
h(idx)
compiler: information not available

C:\Temp>
```

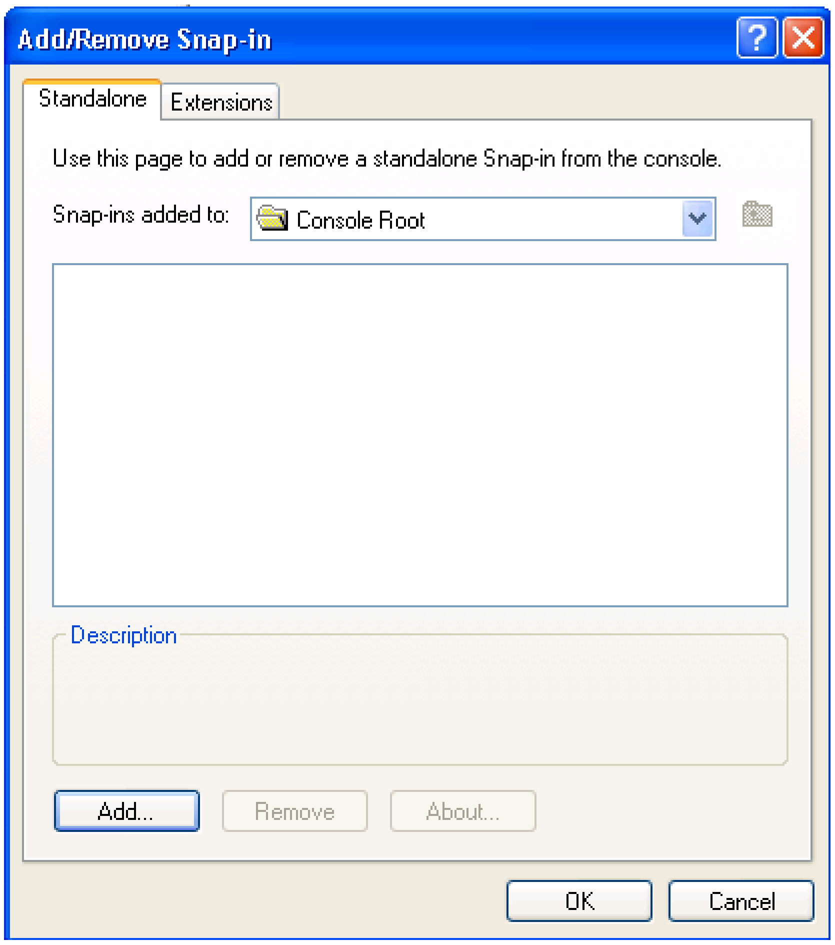
PKCS12

This utility will create (PKCS#12) Personal Information Exchange (.pfx) files mostly used with Microsoft tools such as Internet Information Server. If you have ever gone through the process of getting an SSL certificate from Verisign or Thawte, then this is the certificate you will likely receive. You may therefore utilize the PKCS12 application to generate an SSL certificate for development purposes, remembering that the world according to most browsers will not accept your certificate unless it has been "signed" by one of the root certificates in your CA Trusted Roots list. To see the trusted roots in Windows go to Start | Run and type in "mmc" and press enter. This should bring up the Microsoft Management Console as seen in the next screen shot:



There are no items to show in this view.

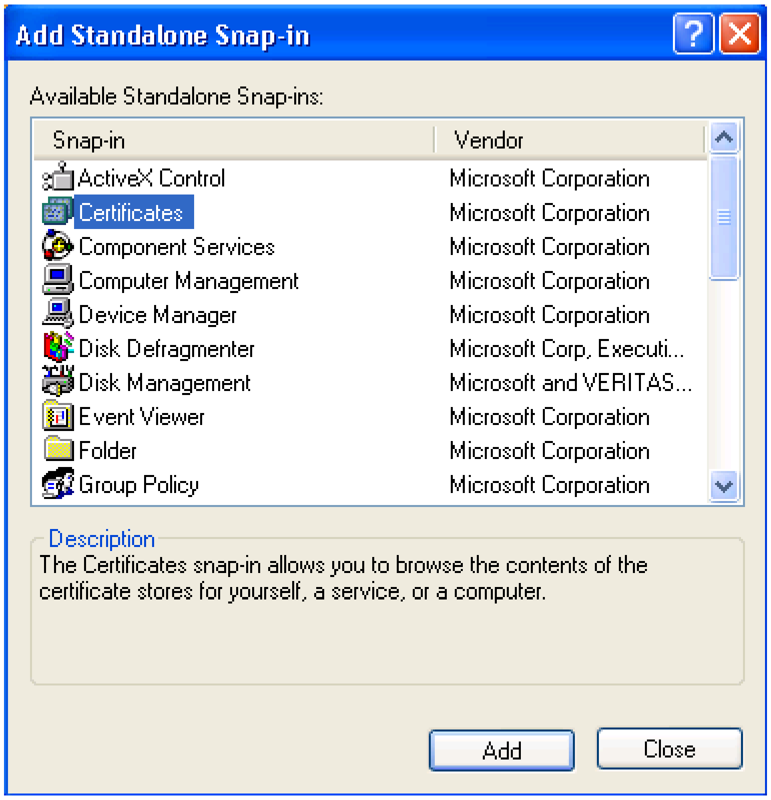
If you don't see exactly what is shown above, don't worry, screens will differ depending on the number of "snap-ins" you have loaded. To see the Trusted Root Certificates using the MMC select File | Add/Remove snap-in. The following dialog should appear:



Use this page to add or remove a standalone Snap-in from the console.

Snap-ins added to: Console Root

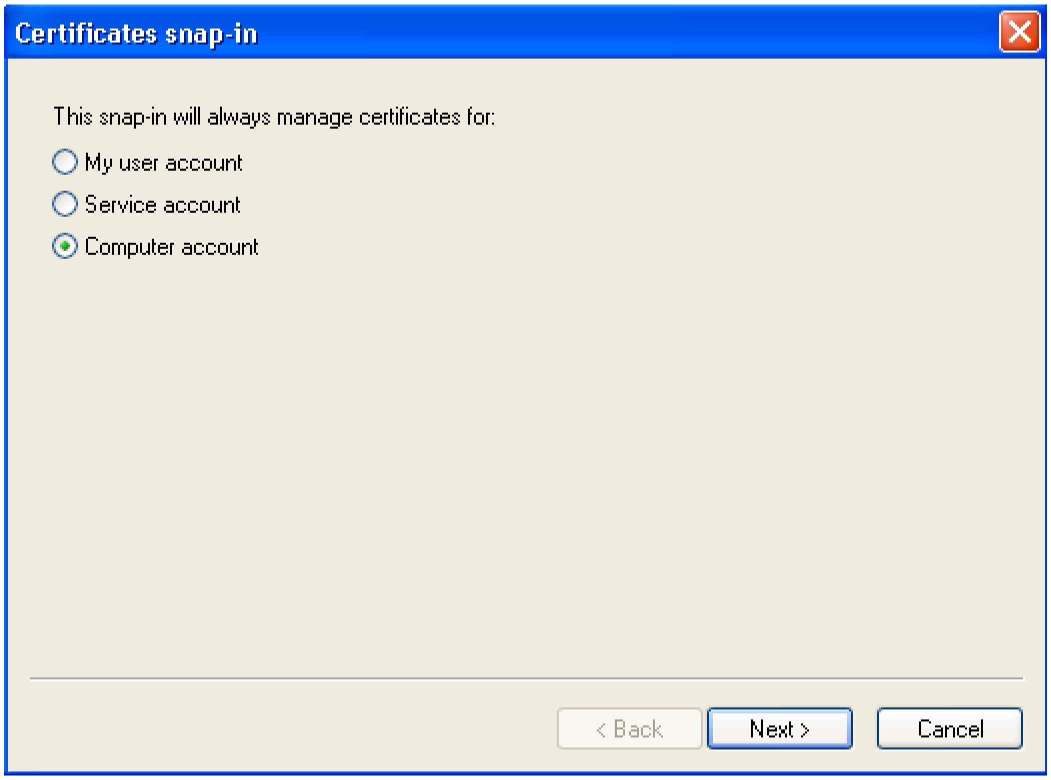
Press the Add… button in the bottom right corner to bring up the next dialog box as shown in the next screen shot:



Snap-in	Vendor
ActiveX Control	Microsoft Corporation
Certificates	Microsoft Corporation
Component Services	Microsoft Corporation
Computer Management	Microsoft Corporation
Device Manager	Microsoft Corporation
Disk Defragmenter	Microsoft Corp, Executi...
Disk Management	Microsoft and VERITAS...
Event Viewer	Microsoft Corporation
Folder	Microsoft Corporation
Group Policy	Microsoft Corporation

Description
The Certificates snap-in allows you to browse the contents of the certificate stores for yourself, a service, or a computer.

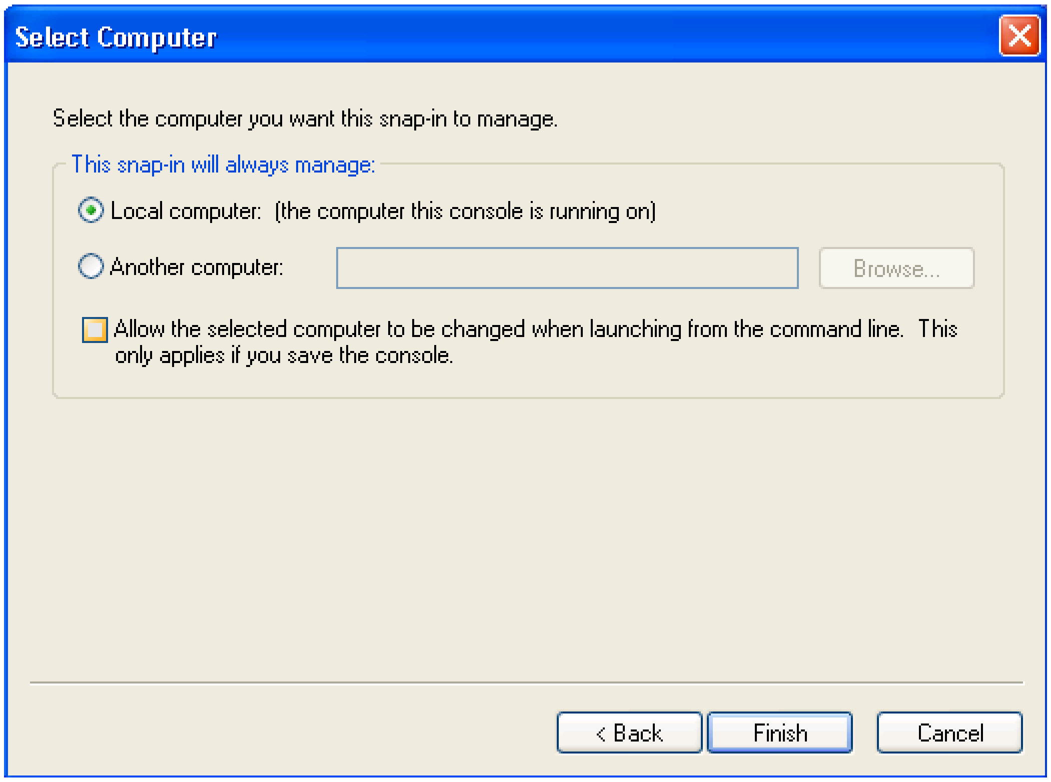
From this dialog, navigate to the Certificates Snap and then press the Add button. You should now see the following dialog:



This snap-in will always manage certificates for:

- ○ My user account
- ○ Service account
- [✓ Computer account]

In the previous dialog make sure you select "Computer account" which will provide you all the certificates and lists for this computer, not just a user. Press the Next > button after making this selection. You should now see the following dialog:



Select Computer

Select the computer you want this snap-in to manage.

This snap-in will always manage:

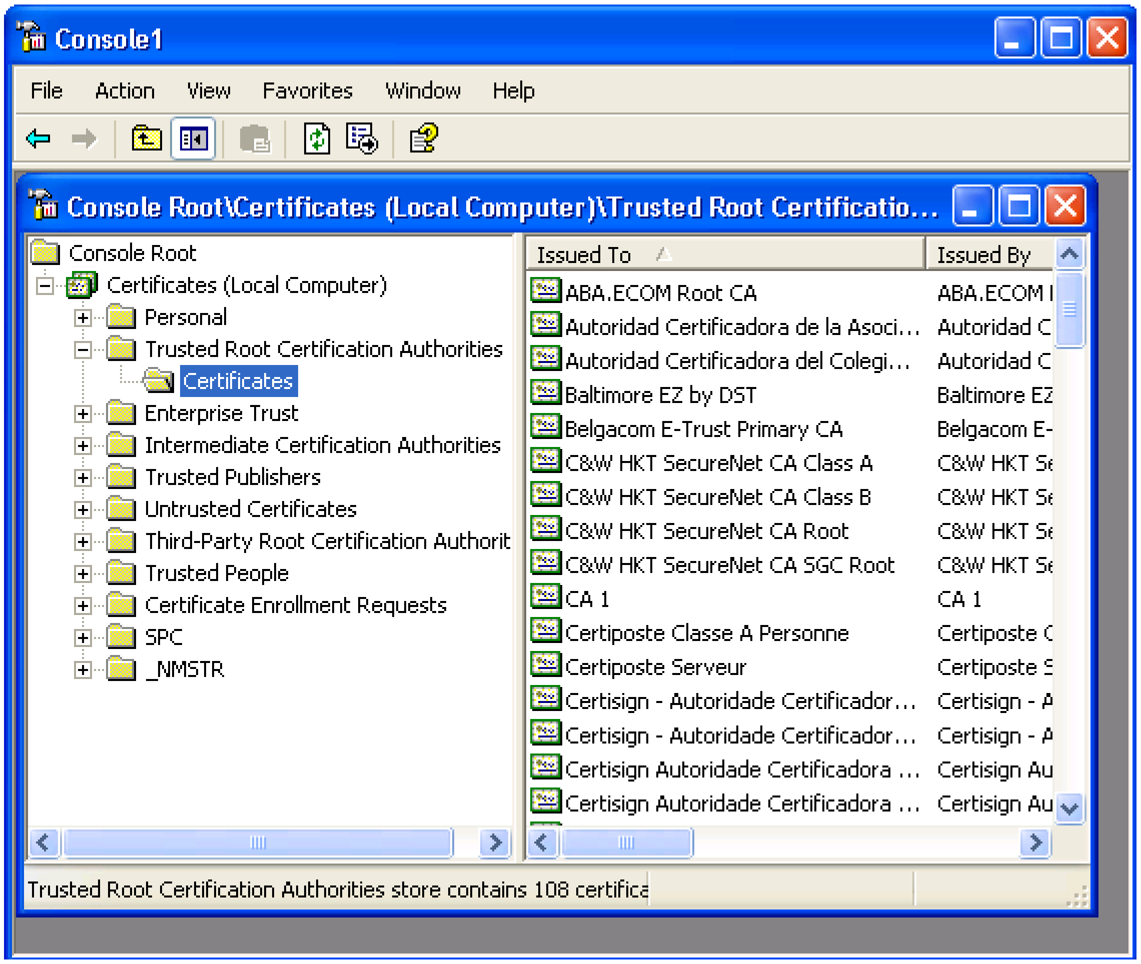
[✓ Local computer: (the computer this console is running on)]

Another computer:

Allow the selected computer to be changed when launching from the command line. This only applies if you save the console.

Make sure you select "Local computer: (the computer this console is running on)" and press the Finish button. This will take you to a prior screen at which point you should press the "Close" button or "Ok" button on all windows until you are back at the main Microsoft Management Console.

In the main MMC (Microsoft Management Console) you should now have the Certificates snap-in displayed in the left pane of the window. Your screen should now look something similar to the following. Pay close attention to the Trusted Root Certificate Authorities:


Console1
File
Action
View
Favorites
Window
Help
Console Root\Certificates (Local Computer)\Trusted Root Certificatio...
Console Root
Certificates (Local Computer)
Personal
Trusted Root Certification Authorities
Certificates
Enterprise Trust
Intermediate Certification Authorities
Trusted Publishers
Untrusted Certificates
Third-Party Root Certification Authorit
Trusted People
Certificate Enrollment Requests
SPC
_NMSTR

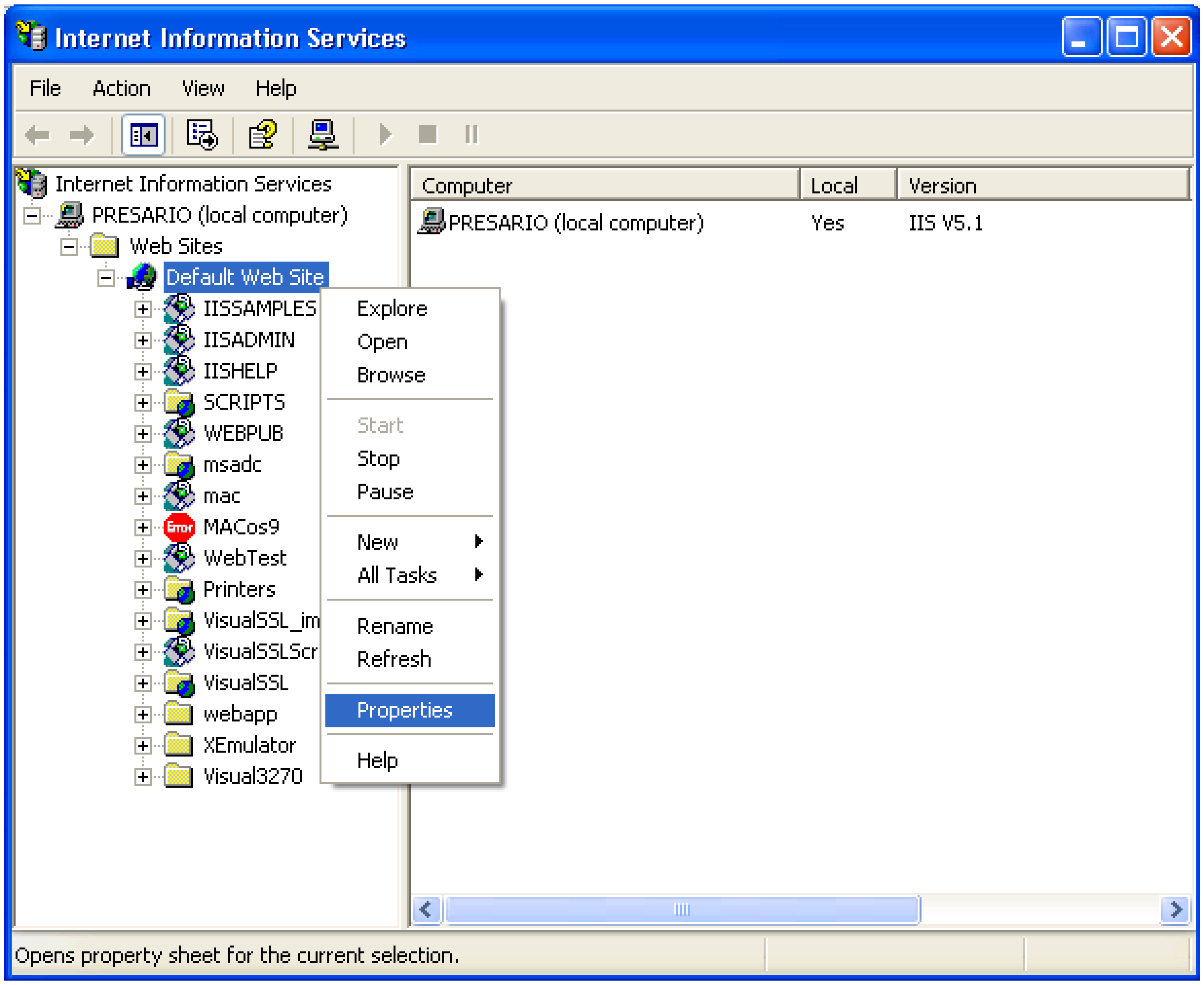
Issued To	Issued By
ABA.ECOM Root CA	ABA.ECOM I
Autoridad Certificadora de la Asoci...	Autoridad C
Autoridad Certificadora del Colegi...	Autoridad C
Baltimore EZ by DST	Baltimore EZ
Belgacom E-Trust Primary CA	Belgacom E-
C&W HKT SecureNet CA Class A	C&W HKT Se
C&W HKT SecureNet CA Class B	C&W HKT Se
C&W HKT SecureNet CA Root	C&W HKT Se
C&W HKT SecureNet CA SGC Root	C&W HKT Se
CA 1	CA 1
Certiposte Classe A Personne	Certiposte C
Certiposte Serveur	Certiposte S
Certisign - Autoridade Certificador...	Certisign - A
Certisign - Autoridade Certificador...	Certisign - A
Certisign Autoridade Certificadora ...	Certisign Au
Certisign Autoridade Certificadora ...	Certisign Au

Trusted Root Certification Authorities store contains 108 certifica

In the above right side pane notice that there are 108 certificates that this computer will trust. If you want to add your certificate to this list, right mouse click on the Certificates folder and select All Tasks | Import and go through the same process as shown earlier in section **Installing the CA Certificate into the Windows Operating System** making sure you are importing the proper format type of your certificate. For the PKCS#12 certificate format continue with this section.

Creating PKCS#12 Certificates in Internet Information Server

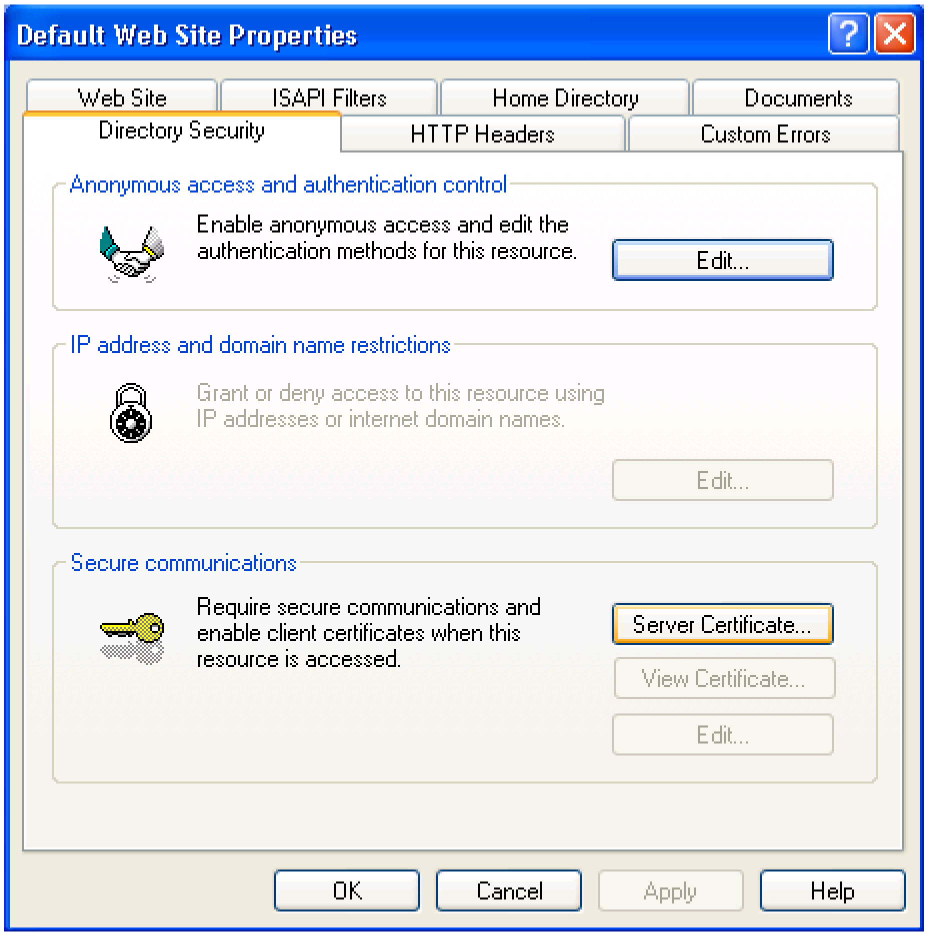
This part is the same process that will be utilized to create a Certificate Signing Request (CSR) that you would send to a Certificate Authority such as Verisign or Thawte. To begin with you must make sure you Internet Information Server 5.0 which comes with Windows 2000 or XP. Windows NT4 users can usually follow the same steps but it may be slightly different. To begin with go to Start | Control Panel and double click on the Administrative Tools. In the Administrative Tools windows select Internet Information Services which will bring up the MMC with the IIS snap-in already loaded. Navigate to the Default Web Site in the left pane and right mouse click on the icon which should bring up the following pop up menu:



Computer	Local	Version
PRESARIO (local computer)	Yes	IIS V5.1

Opens property sheet for the current selection.

When you select the properties for the Default Web Site you should be presented with a tabbed dialog of the options for this site. Windows Workstations should note that they are limited to only the default web site whereas Windows Servers can have several web sites, of which one is usually an SSL enabled site utilizing a PKCS#12 certificate! Your screen, after selecting the Properties option, should look similar to the following:


Default Web Site Properties

Anonymous access and authentication control

Enable anonymous access and edit the authentication methods for this resource.

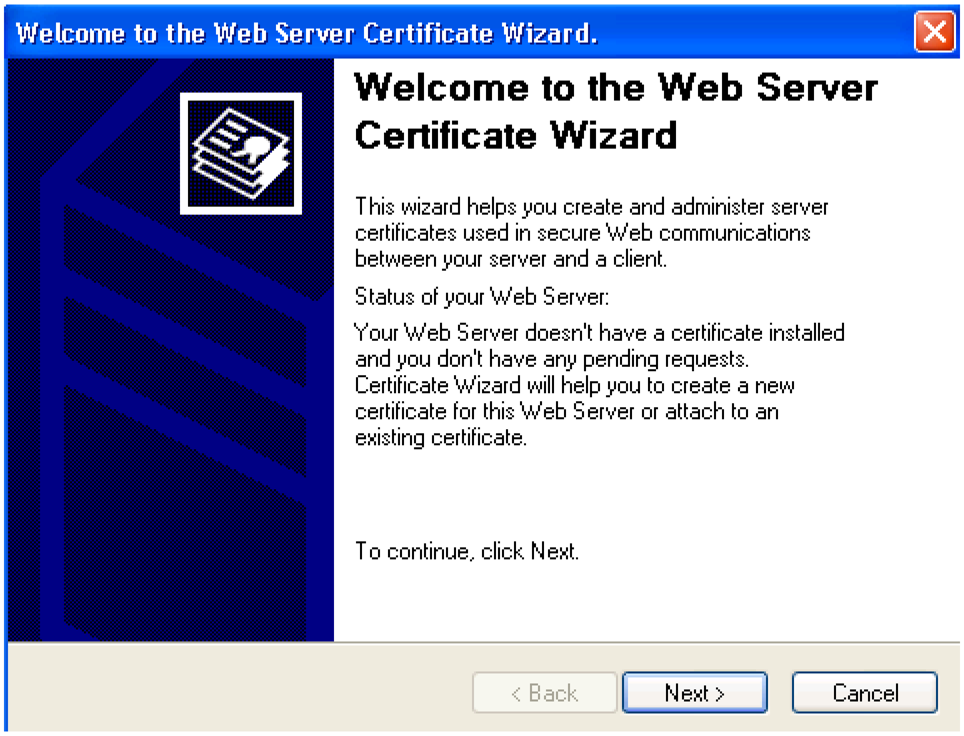
IP address and domain name restrictions

Grant or deny access to this resource using IP addresses or internet domain names.

Secure communications

Require secure communications and enable client certificates when this resource is accessed.

In this above dialog make sure you have selected the Directory Security tab and then the Server Certificate button. You should see the following dialog:


Welcome to the Web Server Certificate Wizard.

Welcome to the Web Server Certificate Wizard

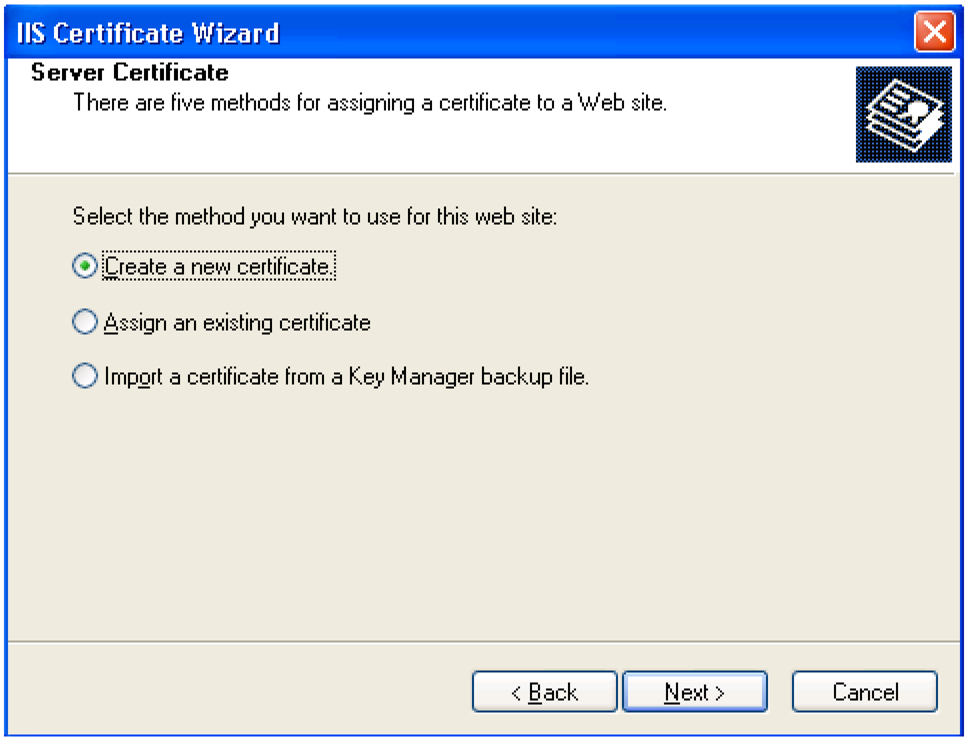
This wizard helps you create and administer server certificates used in secure Web communications between your server and a client.

Status of your Web Server:

Your Web Server doesn't have a certificate installed and you don't have any pending requests. Certificate Wizard will help you to create a new certificate for this Web Server or attach to an existing certificate.

To continue, click Next.

Press the Next > button on the previous screen shot if you have not already done so. You should see the next dialog. Select "Create a new certificate" and press the Next > button.



IIS Certificate Wizard

Server Certificate

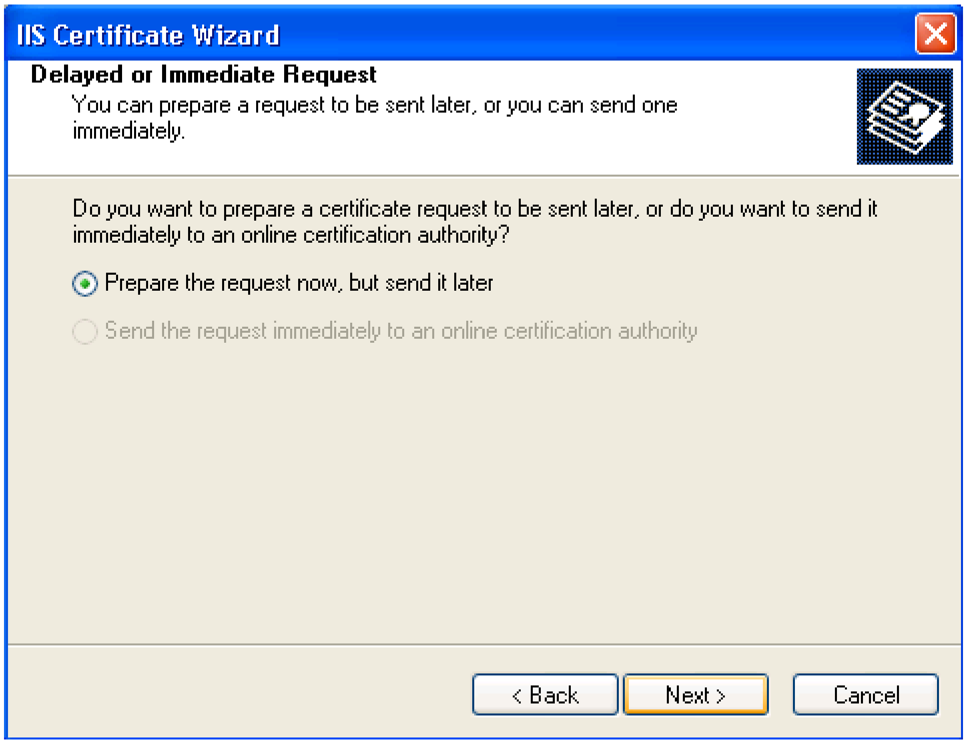
There are five methods for assigning a certificate to a Web site.

Select the method you want to use for this web site:

[✓ Create a new certificate.]

Assign an existing certificate

Import a certificate from a Key Manager backup file.



IIS Certificate Wizard

Delayed or Immediate Request

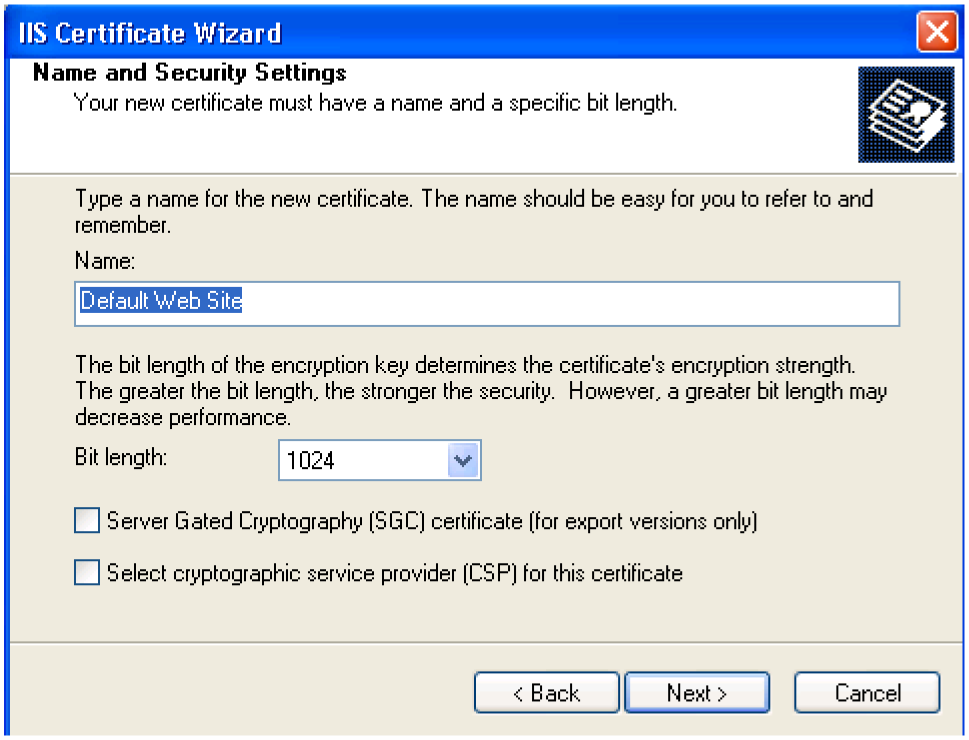
You can prepare a request to be sent later, or you can send one immediately.

Do you want to prepare a certificate request to be sent later, or do you want to send it immediately to an online certification authority?

[✓ Prepare the request now, but send it later]

Send the request immediately to an online certification authority

Select the "Prepare the request now, but send it later" option to create a Public Key pair and its associated Certificate Signing Request (CSR). You won't be able to view the key and public certificate immediately, but you will be able to view the CSR when you complete the whole process. We will discuss exporting the public key pair later in this section. If you have not already done so, press the Next > button:



IIS Certificate Wizard

Name and Security Settings

Your new certificate must have a name and a specific bit length.

Type a name for the new certificate. The name should be easy for you to refer to and remember.

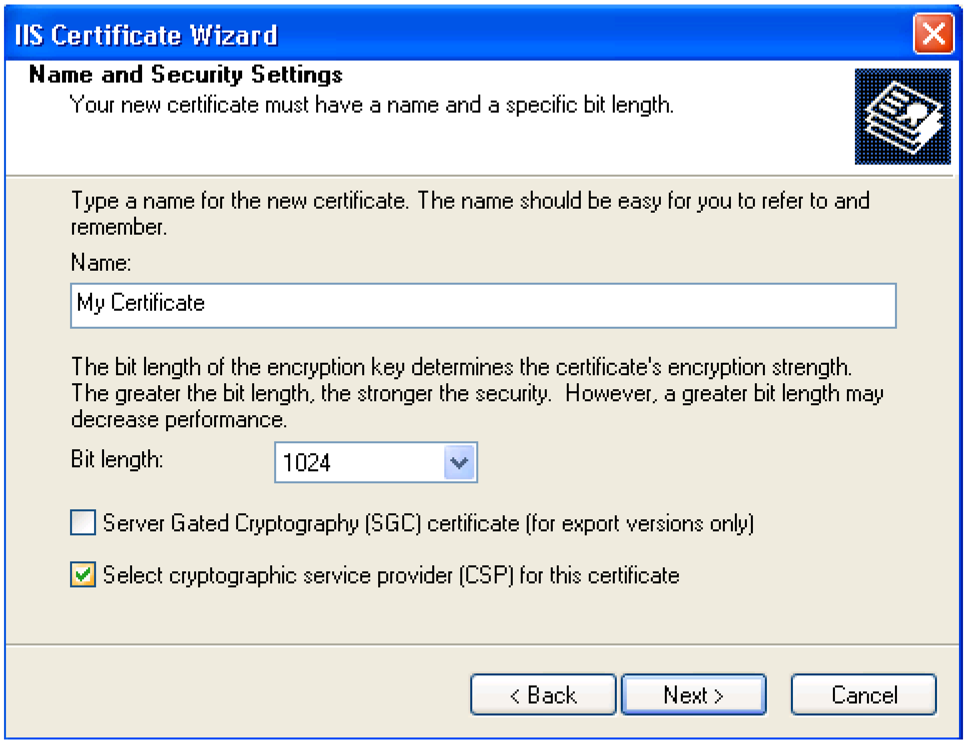
Name: Default Web Site

The bit length of the encryption key determines the certificate's encryption strength. The greater the bit length, the stronger the security. However, a greater bit length may decrease performance.

Bit length: 1024

☐ Server Gated Cryptography (SGC) certificate (for export versions only)

☐ Select cryptographic service provider (CSP) for this certificate



IIS Certificate Wizard

Name and Security Settings

Your new certificate must have a name and a specific bit length.

Type a name for the new certificate. The name should be easy for you to refer to and remember.

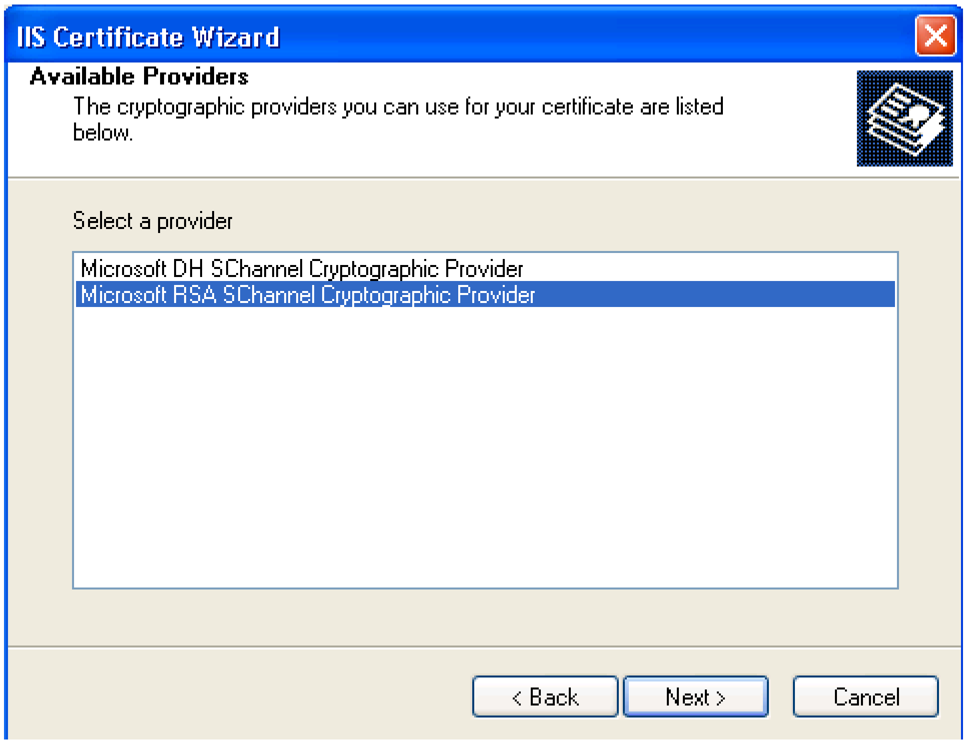
Name: My Certificate

The bit length of the encryption key determines the certificate's encryption strength. The greater the bit length, the stronger the security. However, a greater bit length may decrease performance.

Bit length: 1024

☐ Server Gated Cryptography (SGC) certificate (for export versions only)

☑ Select cryptographic service provider (CSP) for this certificate


IIS Certificate Wizard

Available Providers

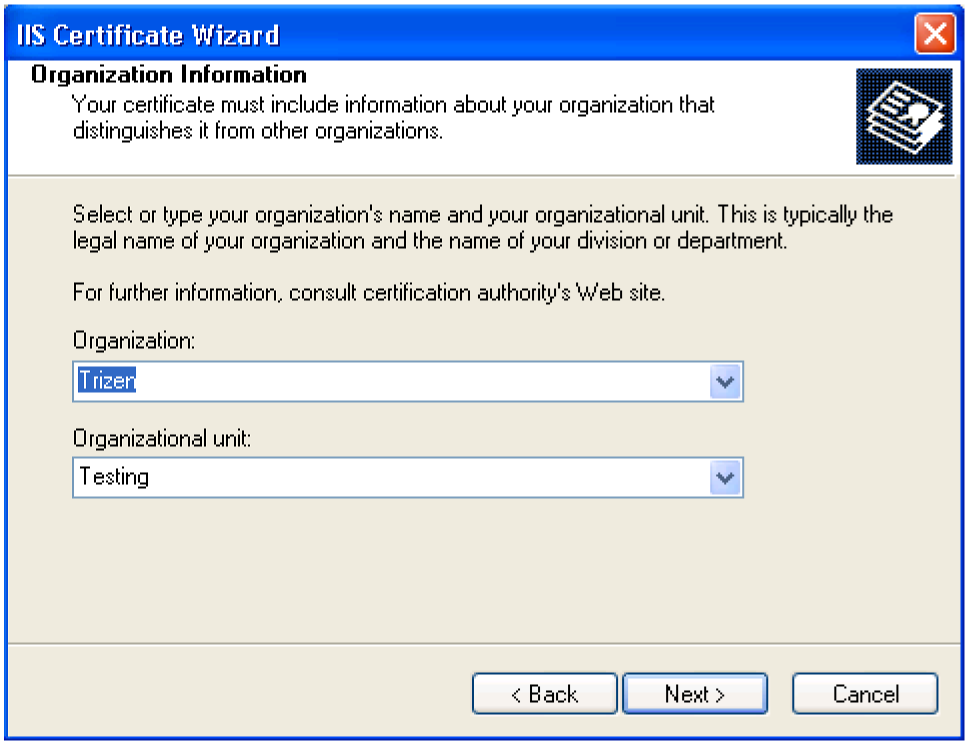
The cryptographic providers you can use for your certificate are listed below.

Select a provider

Microsoft DH SChannel Cryptographic Provider
Microsoft RSA SChannel Cryptographic Provider

In the above dialogs select an easy to remember name with a bit length of 1024. If you have created a Public Key pair earlier using OpenSSL, this looks very familiar. Make sure you select the "Select cryptographic service provider (CSP) for this certificate." This will be important if using Authenticode and other technologies. After pressing the Next > button, make sure you select Microsoft RSA Schannel Cryptographic Provider and press the Next > button.

Schannel is Microsoft's answer to OpenSSL without the source code and open environment. This is commonly referred to as the CyptoAPI by Microsoft and is almost always found as a Dynamic Link Library.


IIS Certificate Wizard

Organization Information

Your certificate must include information about your organization that distinguishes it from other organizations.

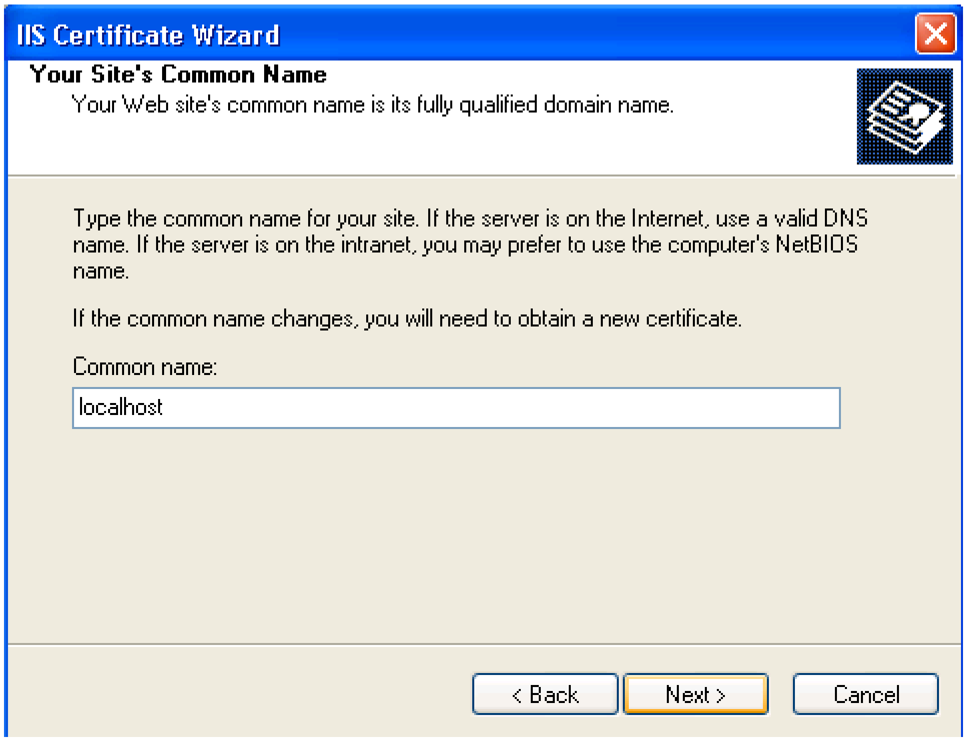
Select or type your organization's name and your organizational unit. This is typically the legal name of your organization and the name of your division or department.

For further information, consult certification authority's Web site.

Organization: Trizen

Organizational unit: Testing

Add your information in the edit boxes provided in the above dialog. Don't worry if you make a mistake, you can always go back. Its only important when you are ready to send this information to a Certificate Authority.


IIS Certificate Wizard

Your Site's Common Name

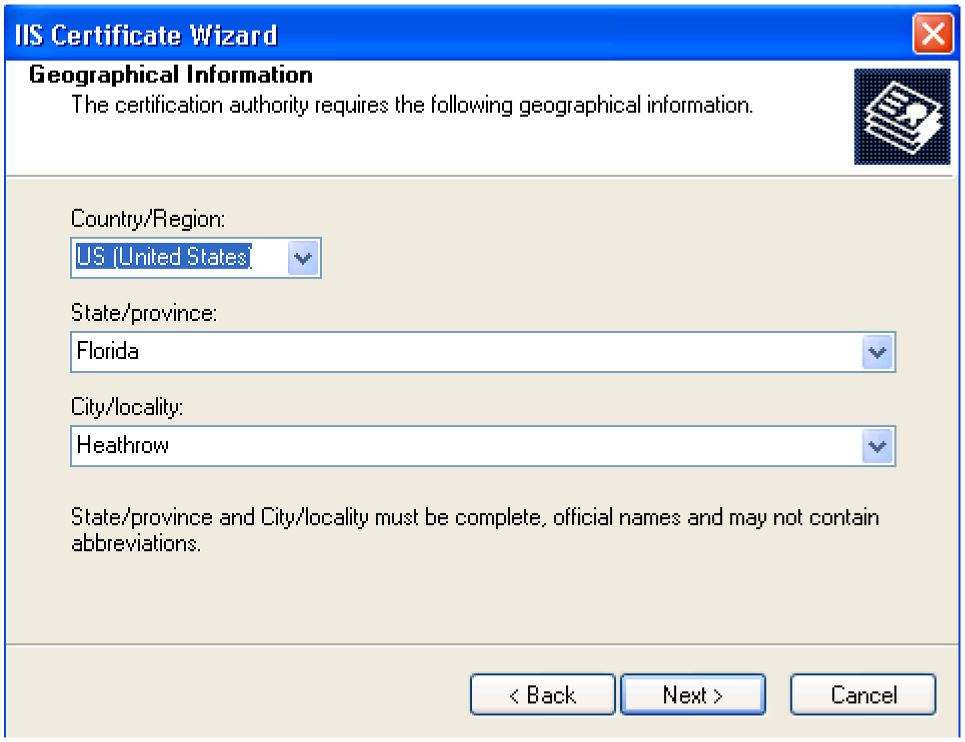
Your Web site's common name is its fully qualified domain name.

Type the common name for your site. If the server is on the Internet, use a valid DNS name. If the server is on the intranet, you may prefer to use the computer's NetBIOS name.

If the common name changes, you will need to obtain a new certificate.

Common name: localhost

In the above dialog box place in your fully qualified domain name such as www.trizen.com or if you're only interested in testing, use localhost. Press the Next > button to complete the information regarding your certificate.



Geographical Information

The certification authority requires the following geographical information.

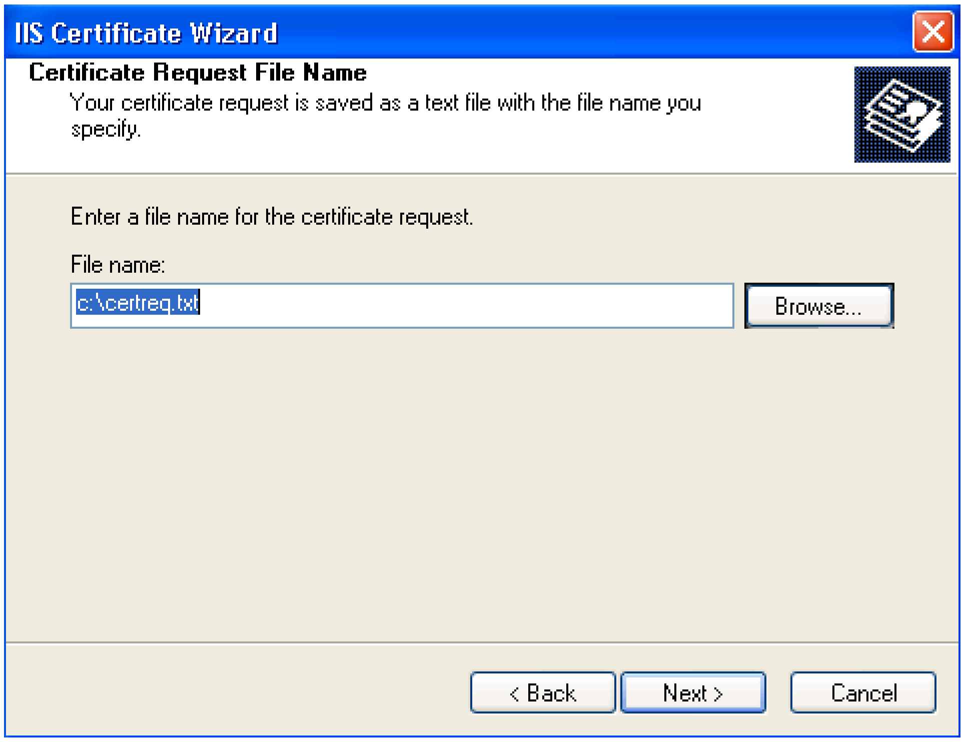
Country/Region: US (United States)

State/province: Florida

City/locality: Heathrow

State/province and City/locality must be complete, official names and may not contain abbreviations.

After filling in your information press the Next > button:



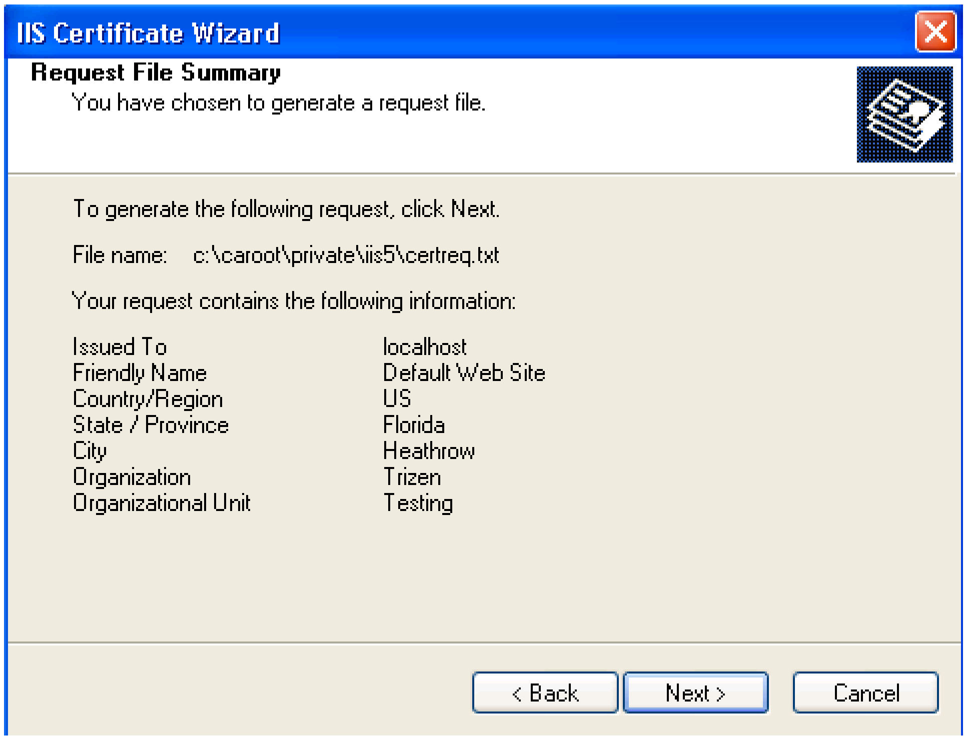
Certificate Request File Name

Your certificate request is saved as a text file with the file name you specify.

Enter a file name for the certificate request.

File name: c:\certreq.txt

Select the filename and directory location where you would like to save your Certificate Signing Request. We typically save them near or in the CARoot directory created at the beginning of this document. When you press the Next > button you should see the following dialog:



Request File Summary

You have chosen to generate a request file.

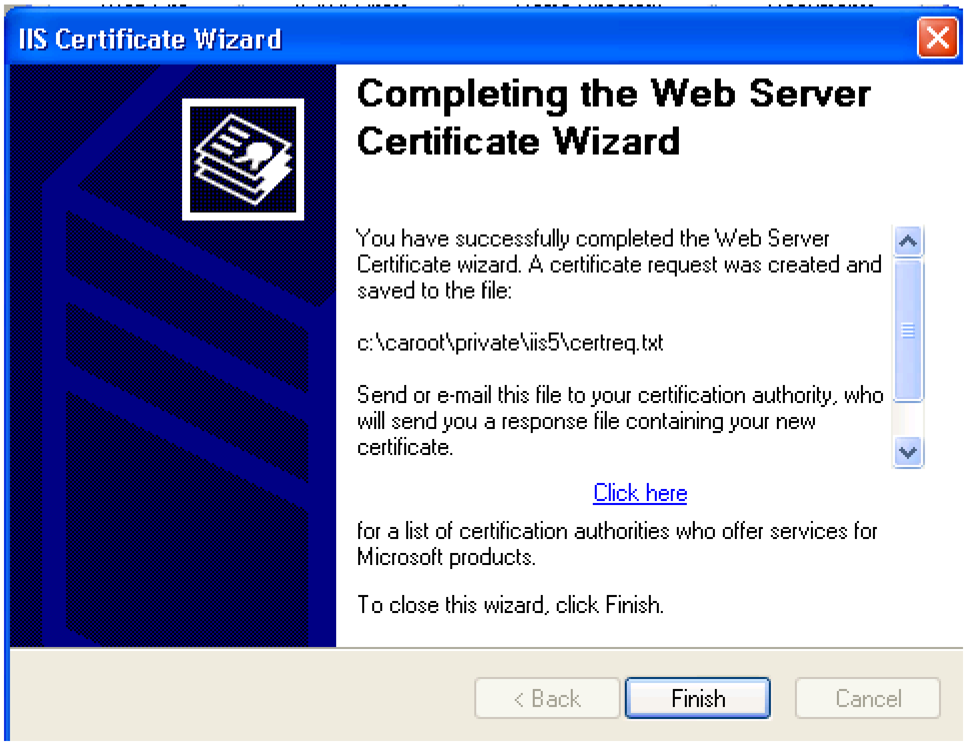
To generate the following request, click Next.

File name: c:\caroot\private\iis5\certreq.txt

Your request contains the following information:

Issued To	localhost
Friendly Name	Default Web Site
Country/Region	US
State / Province	Florida
City	Heathrow
Organization	Trizen
Organizational Unit	Testing

Press the Next > button to complete the certificate creation process.



Completing the Web Server Certificate Wizard

You have successfully completed the Web Server Certificate wizard. A certificate request was created and saved to the file:

c:\caroot\private\iis5\certreq.txt

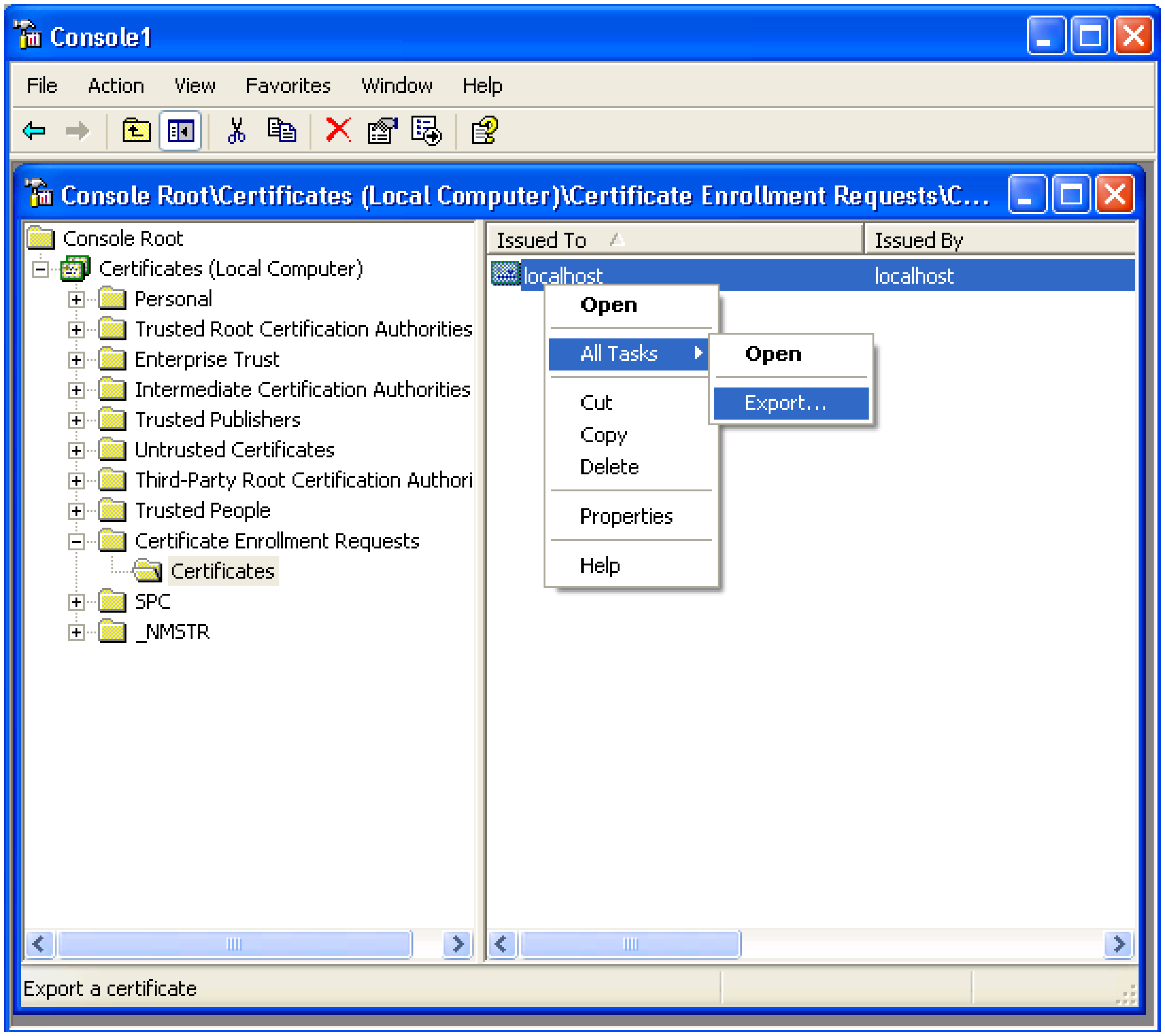
Send or e-mail this file to your certification authority, who will send you a response file containing your new certificate.

Click here

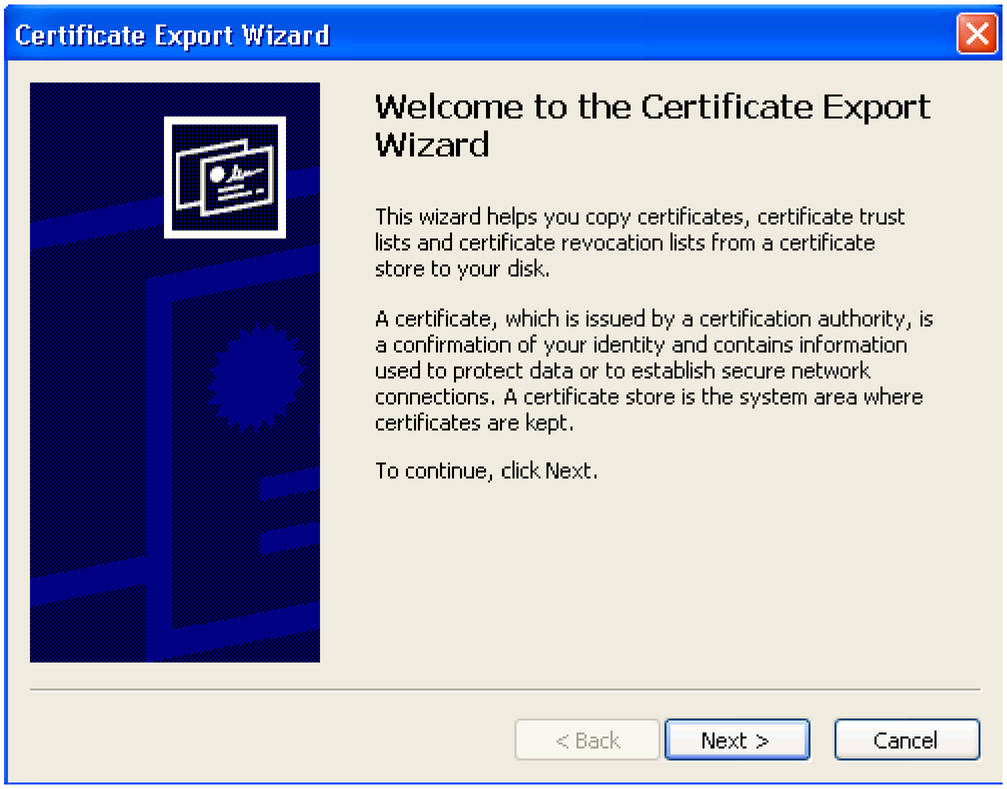
for a list of certification authorities who offer services for Microsoft products.

To close this wizard, click Finish.

Once you press the Finish button on the previous dialog, you should be taken back to the tabbed properties dialog for the Default Web Site. Press Cancel to return to the Internet Information Services Management Console. So where is the certificate and key file? It has been captured into the Microsoft Certificate's snap in. If you recall from the earlier discussion on locating all the Trusted Root Certificate Authorities by using the Certificates snap-in then you can easily export the certificate and key file from this utility as shown below:



Notice that the certificate we just created is contained in the Certificate Enrollment Requests | Certificates files. Just right mouse click on the actual certificate in the right pane and select Export… to actually export the certificate.



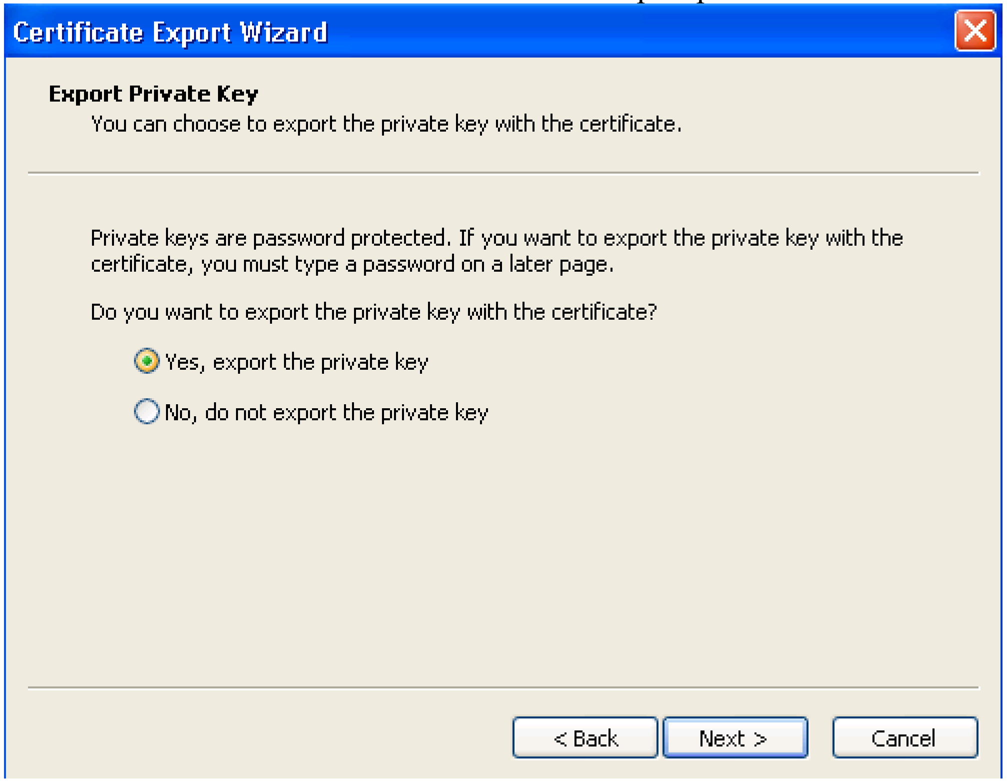
Welcome to the Certificate Export Wizard

This wizard helps you copy certificates, certificate trust lists and certificate revocation lists from a certificate store to your disk.

A certificate, which is issued by a certification authority, is a confirmation of your identity and contains information used to protect data or to establish secure network connections. A certificate store is the system area where certificates are kept.

To continue, click Next.

Press the Next > button to continue the export process.



Export Private Key

You can choose to export the private key with the certificate.

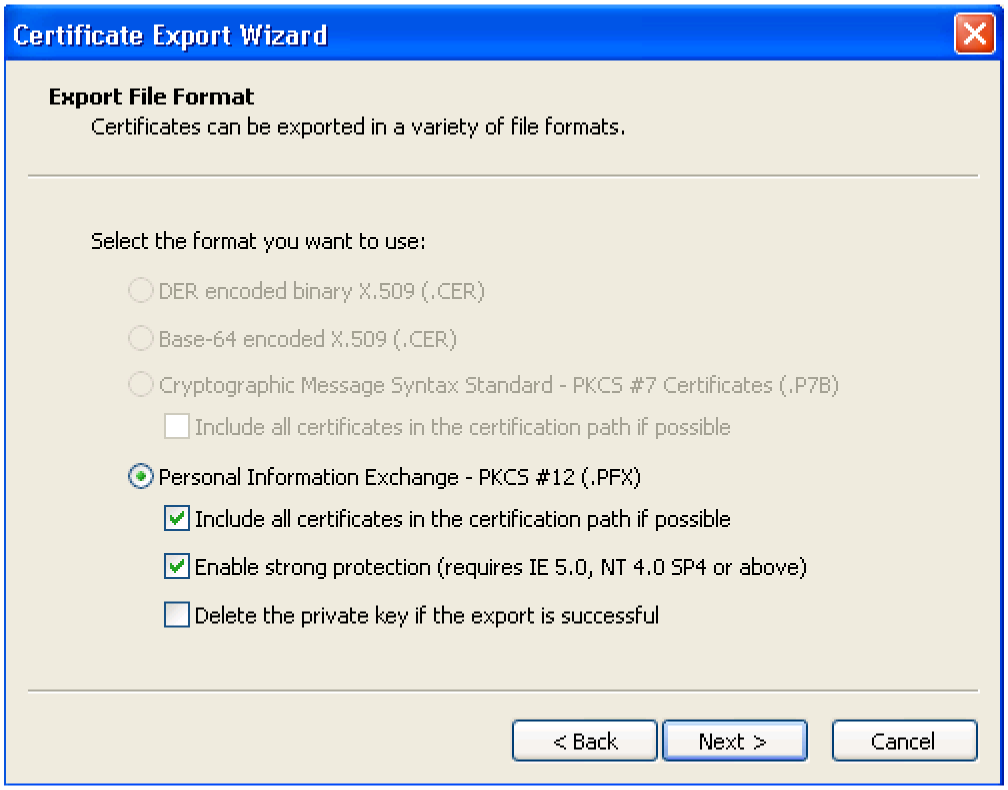
Private keys are password protected. If you want to export the private key with the certificate, you must type a password on a later page.

Do you want to export the private key with the certificate?

[✓ Yes, export the private key]

No, do not export the private key

Make sure you select “Yes, export the private key” option.

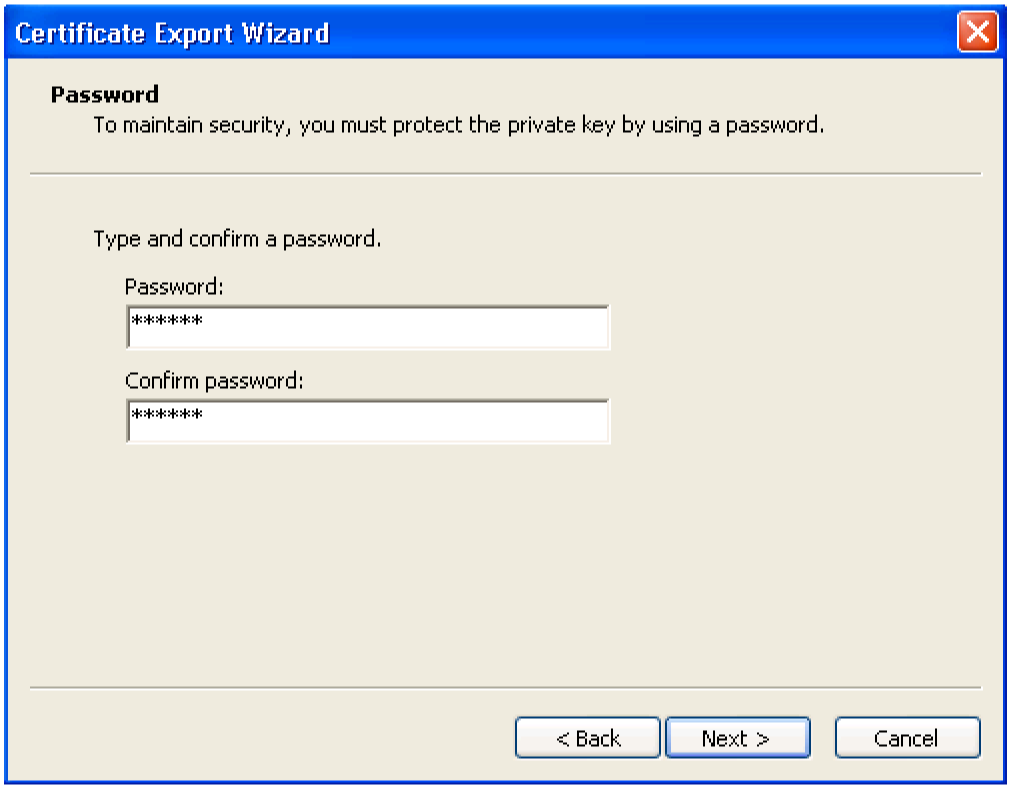


Export File Format
Certificates can be exported in a variety of file formats.

Select the format you want to use:

- ○ DER encoded binary X.509 (.CER)
- ○ Base-64 encoded X.509 (.CER)
- ○ Cryptographic Message Syntax Standard - PKCS #7 Certificates (.P7B)
  - ☐ Include all certificates in the certification path if possible
- ● Personal Information Exchange - PKCS #12 (.PFX)
  - ✓ Include all certificates in the certification path if possible
  - ✓ Enable strong protection (requires IE 5.0, NT 4.0 SP4 or above)
  - ☐ Delete the private key if the export is successful

Make sure you select the same options as above.



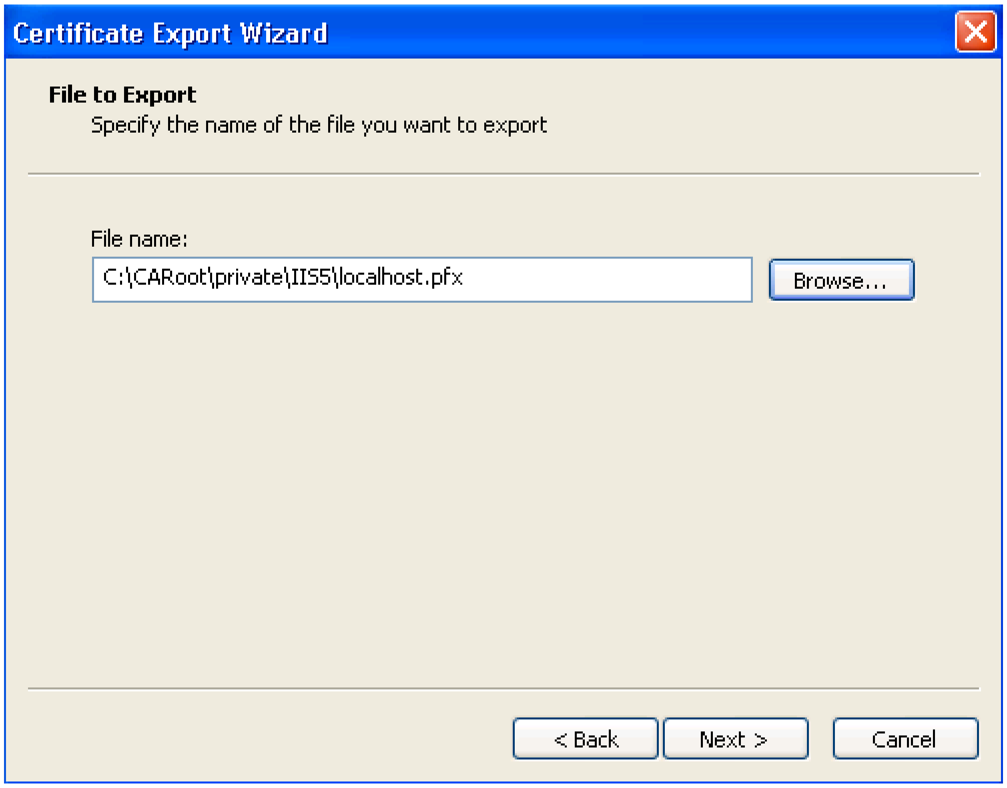
Password
To maintain security, you must protect the private key by using a password.

Type and confirm a password.

Password: ******

Confirm password: ******

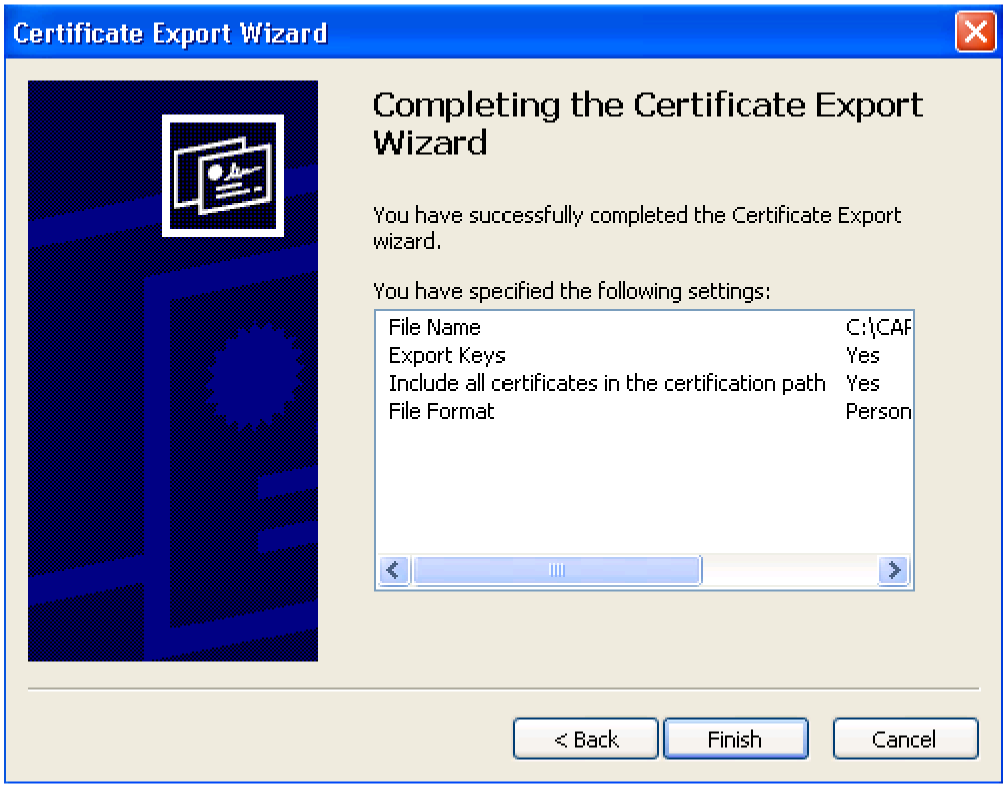
Type a password for your key file making sure you don't forget it!



File to Export
Specify the name of the file you want to export

File name:
C:\CARoot\private\IIS5\localhost.pfx

Select the filename for the key that will be exported as well as its location and press the Next > button. On the next screen just press Finish to complete the process.



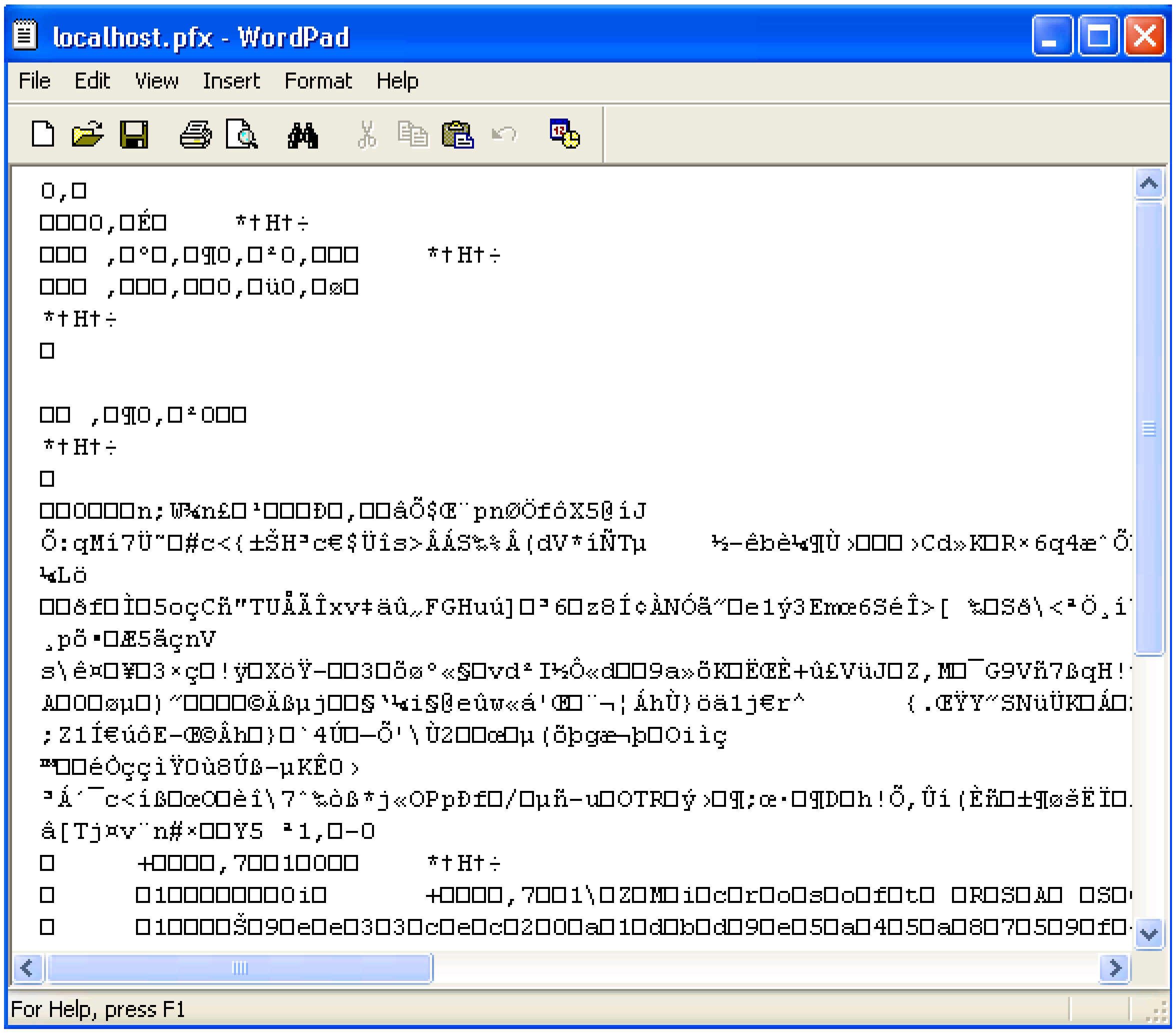
Completing the Certificate Export Wizard

You have successfully completed the Certificate Export wizard.

You have specified the following settings:

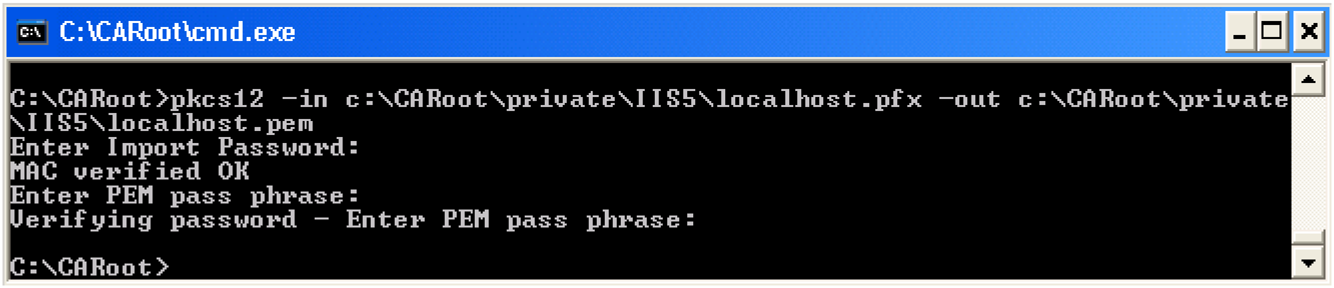
File Name	C:\CAR
Export Keys	Yes
Include all certificates in the certification path	Yes
File Format	Person

At this point you have created a .pfx file that should reside in the directory you selected when you exported the file. We highly suggest you export these files near your CARoot directory structure. If you merely double click on the newly created .pfx file Windows will bring up the certificate import wizard, just press cancel if this pops up because what you really want is to look at the certificate and the key file. If you open the file up in WordPad you should see a lot of encrypted information:



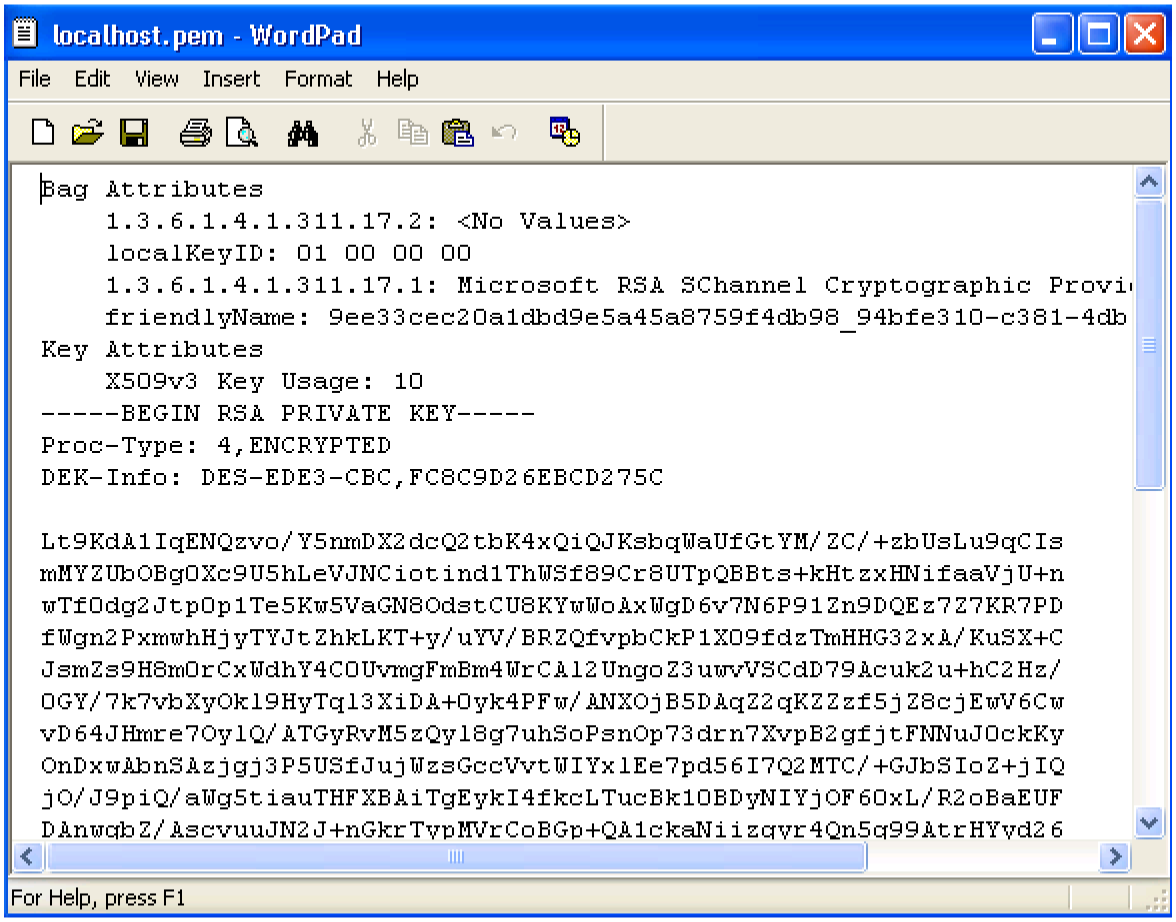
```
0,□
□□□0,□É□      *†Ht÷
□□□ ,□°□,□¶0,□²0,□□□      *†Ht÷
□□□ ,□□□,□□0,□ü0,□ø□
*†Ht÷
□

□□ ,□¶0,□²0□□
*†Ht÷
□
□□0□□□n;W¾n£□¹□□□Ð□,□□âÕ$Œ¨pnØÖfôX5@iJ
Õ:qMi7Ü˜□#c<(±ŠH³c€$Üis>ÂÁS‰%Â(dV*iÑTµ      ½–êbè¼¶Ù›□□□›Cd»K□R×6q4æˆÕ
¼Lö
□□ðf□Ì□5oçCñ"TUÅÃÎxv‡äû„FGHuú]□³6□z8Í¢ÀNÓã˝□e1ý3Emœ6SéÎ>[ ‰□Sð\<²Ö¸í
¸põ•□Æ5ãçnV
s\ê¤□¥□3×ç□!ÿ□XöŸ–□□3□õø°«§□vd²I½Ô«d□□9a»õK□ËŒÈ+û£VüJ□Z,M□¯G9Vñ7ßqH!
A□O□øµ□)˝□□□□©ÄßµjO□§‘¼i§@eûw«á'Œ□¨¬¦ÁhÙ}öä1j€r^        {.ŒŸY˝SNüÜK□Á□
;Z1Í€úôE–Œ®Âh□}□`4Ú□—Õ'\Ù2□□œ□µ(õþgæ¬þ□Oiiç
™□□éÒççiŸOù8Úß–µKÊO›
³Á´¯c<íß□œO□èî\7ˆ‰òß*j«OPpÐf□/□µñ–u□OTR□ý›□¶;œ·□¶D□h!Õ,Ûi(Èñ□±¶øšËÏ□
â[Tj¤v¨n#×□□Y5 ²1,□–0
□      +□□□□,7□□1□□□□      *†Ht÷
□      □1□□□□□□□□0i□       +□□□□,7□□1\□Z□M□i□c□r□o□s□o□f□t□ □R□S□A□ □S□
□      □1□□□□Š□9□e□e□3□3□c□e□c□2□0□a□1□d□b□d□9□e□5□a□4□5□a□8□7□5□9□f□
```

The above file is nowhere near human readable and as such needs to be parsed with the PKCS12 utility before the key and certificate can be viewed. If you have not already done so, create a directory named IIS5 in your c:\CARoot\private directory and place this .pfx file into this location. Next, open up the cmd.exe file or command.com file in your CARoot directory and type the following to parse your PKCS#12 certificate:



```
C:\CARoot>pkcs12 -in c:\CARoot\private\IIS5\localhost.pfx -out c:\CARoot\private
\IIS5\localhost.pem
Enter Import Password:
MAC verified OK
Enter PEM pass phrase:
Verifying password - Enter PEM pass phrase:

C:\CARoot>
```

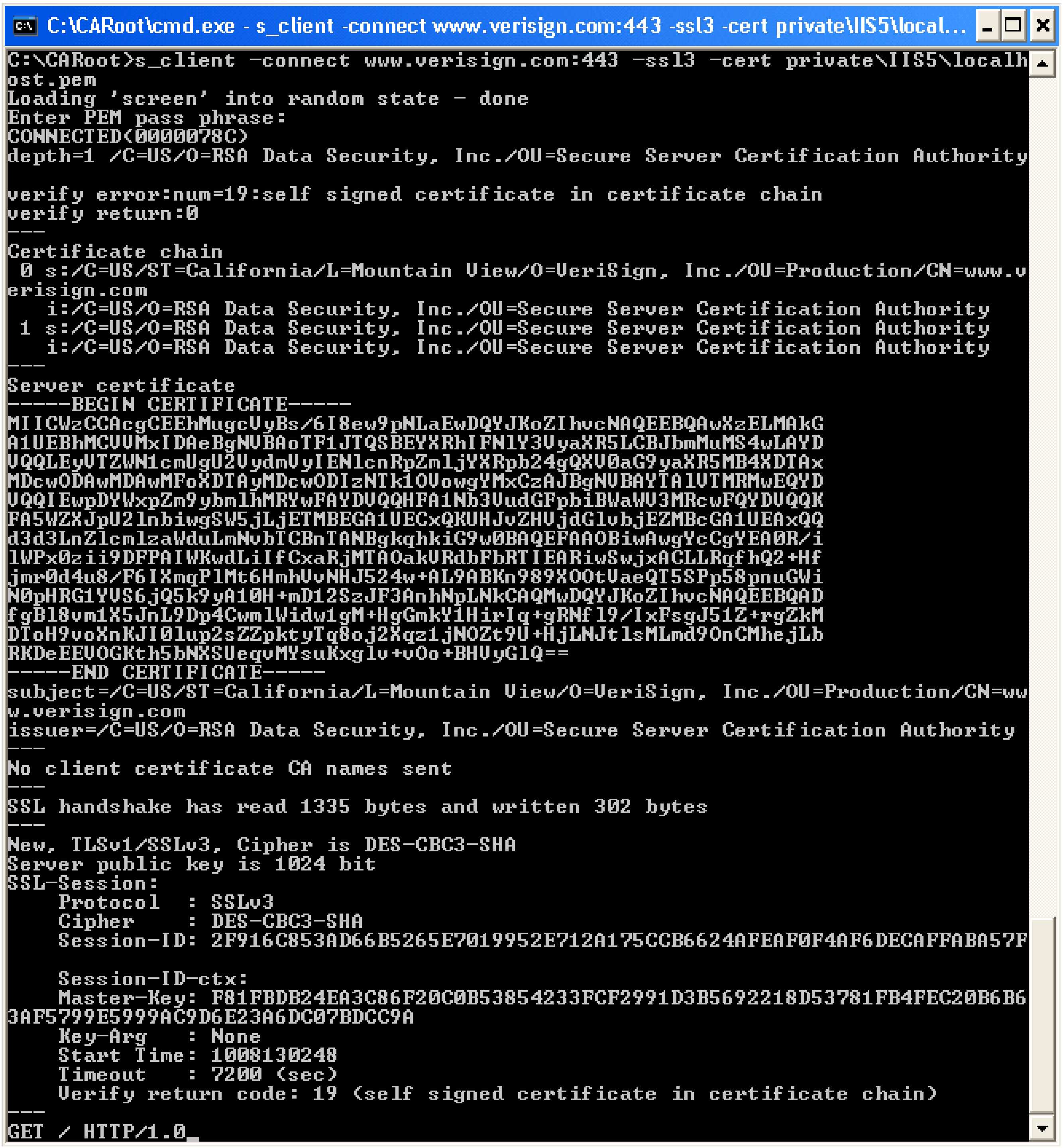
You should notice that we are parsing the localhost.pfx file that was exported earlier in this section. You may have a different directory structure and as such your MS-DOS prompt may look slightly different; however, the above demonstrates the parsing and exporting of a new key and certificate file into one file. In our demonstration we have copied the localhost.pfx file to the c:\CARoot\private\IIS5 directory and exporting the parsed file to the same location with the name localhost.pem. Notice that these files are in PEM format. Before the file can be completely parsed you must of course type in the password for the encrypted file that was requested when you exported the localhost.pfx file. You should keep this password when prompted to enter a password for the localhost.pem file. At this point you can double click on the localhost.pem file:



```
Bag Attributes
    1.3.6.1.4.1.311.17.2: <No Values>
    localKeyID: 01 00 00 00
    1.3.6.1.4.1.311.17.1: Microsoft RSA SChannel Cryptographic Provi
    friendlyName: 9ee33cec20a1dbd9e5a45a8759f4db98_94bfe310-c381-4db
Key Attributes
    X509v3 Key Usage: 10
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: DES-EDE3-CBC,FC8C9D26EBCD275C

Lt9KdA1IqENQzvo/Y5nmDX2dcQ2tbK4xQiQJKsbqWaUfGtYM/ZC/+zbUsLu9qCIs
mMYZUbOBg0Xc9U5hLeVJNCiotind1ThWSf89Cr8UTpQBBts+kHtzxHNifaaVjU+n
wTf0dg2JtpOp1Te5Kw5VaGN8OdstCU8KYwWoAxWgD6v7N6P91Zn9DQEz7Z7KR7PD
fWgn2PxmwhHjyTYJtZhkLKT+y/uYV/BRZQfvpbCkP1XO9fdzTmHHG32xA/KuSX+C
JsmZs9H8mOrCxWdhY4COUvmgFmBm4WrCAl2UngoZ3uwvVSCdD79Acuk2u+hC2Hz/
OGY/7k7vbXyOkl9HyTql3XiDA+Oyk4PFw/ANXOjB5DAqZ2qKZZzf5jZ8cjEwV6Cw
vD64JHmre7Oy1Q/ATGyRvM5zQyl8g7uhSoPsnOp73drn7XvpB2gfjtFNNuJ0ckKy
OnDxwAbnSAzjgj3P5USfJujWzsGccVvtWIYxlEe7pd56I7Q2MTC/+GJbSIoZ+jIQ
jO/J9piQ/aWg5tiauTHFXBAiTgEykI4fkcLTucBk10BDyNIYjOF60xL/R2oBaEUF
DAnwqbZ/AscvuuJN2J+nGkrTvpMVrCoBGp+QA1ckaNiizqvr4Qn5q99AtrHYvd26
```

This file should look somewhat familiar by now and can be used within your OpenSSL applications. OpenSSL applications allow the use of the certificate and the key within the same file as demonstrated by typing in the following into your MS-DOS prompt, assuming you have access to the OpenSSL applications via your PATH variable as discussed early during the install:

```
s_client –connect www.verisign.com:443 -ssl3 –cert c:\CARoot\private\IIS5\localhost.pem
```

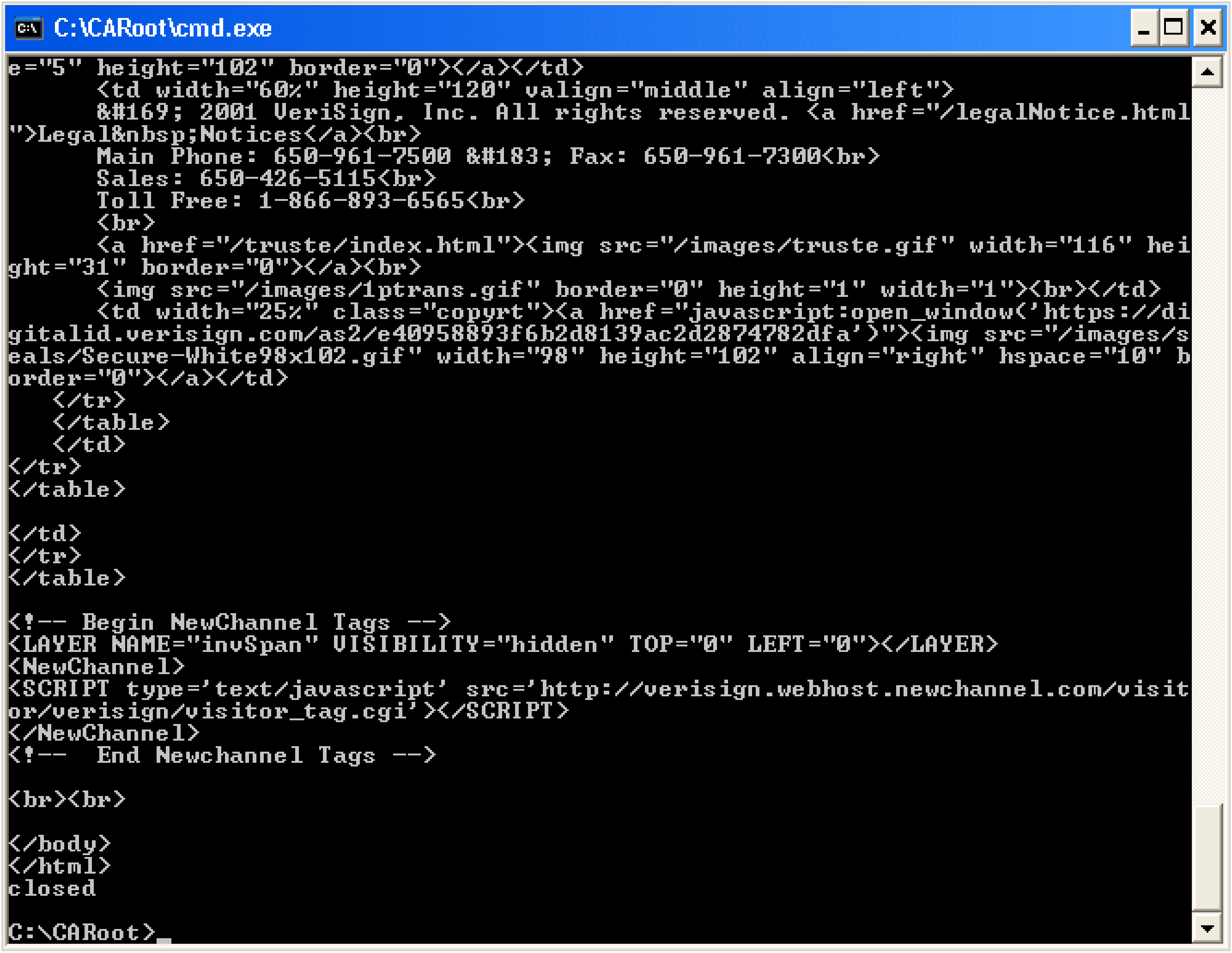

C:\CARoot\cmd.exe - s_client -connect www.verisign.com:443 -ssl3 -cert private\IIS5\local...

```
C:\CARoot>s_client -connect www.verisign.com:443 -ssl3 -cert private\IIS5\localh
ost.pem
Loading 'screen' into random state - done
Enter PEM pass phrase:
CONNECTED(0000078C)
depth=1 /C=US/O=RSA Data Security, Inc./OU=Secure Server Certification Authority

verify error:num=19:self signed certificate in certificate chain
verify return:0
---
Certificate chain
 0 s:/C=US/ST=California/L=Mountain View/O=VeriSign, Inc./OU=Production/CN=www.v
erisign.com
   i:/C=US/O=RSA Data Security, Inc./OU=Secure Server Certification Authority
 1 s:/C=US/O=RSA Data Security, Inc./OU=Secure Server Certification Authority
   i:/C=US/O=RSA Data Security, Inc./OU=Secure Server Certification Authority
---
Server certificate
-----BEGIN CERTIFICATE-----
MIICWzCCAcgCEEhMugcVyBs/6I8ew9pNLaEwDQYJKoZIhvcNAQEEBQAwXzELMAkG
A1UEBhMCVVMxIDAeBgNVBAoTF1JTQSBEYXRhIFN1Y3VyaXR5LCBJbmMuMS4wLAYD
VQQLEyVTZWN1cmUgU2VydmVyIENlcnRpZmljYXRpb24gQXV0aG9yaXR5MB4XDTAx
MDcwODAwMDAwMFoXDTAyMDcwODIzNTk1OVowgYMxCzAJBgNVBAYTAlVTMRMwEQYD
VQQIEwpDYWxpZm9ybmlhMRYwFAYDVQQHFA1Nb3VudGFpbiBWaWV3MRcwFQYDVQQK
FA5WZXJpU2lnbiwgSW5jLjETMBEGA1UECxQKUHJvZHVjdGlvbjEZMBcGA1UEAxQQ
d3d3LnZlcmlzaWduLmNvbTCBnTANBgkqhkiG9w0BAQEFAAOBiwAwgYcCgYEA0R/i
lWPx0zii9DFPAIWKwdLiIfCxaRjMTAOakVRdbFbRTIEARiwSwjxACLLRqfhQ2+Hf
jmr0d4u8/F6IXmqPlMt6HmhVvNHJ524w+AL9ABKn989XOOtVaeQT5SPp58pnuGWi
N0pHRG1YVS6jQ5k9yA10H+mD12SzJF3AnhNpLNkCAQMwDQYJKoZIhvcNAQEEBQAD
fgBl8vm1X5JnL9Dp4CwmlWidw1gM+HgGmkY1HirIq+gRNfl9/IxFsgJ51Z+rgZkM
DToH9voXnKJI0lup2sZZpktyTq8oj2Xqz1jNOZt9U+HjLNJtlsMLmd9OnCMhejLb
RKDeEEVOGKth5bNXSUeqvMYsuKxglv+vOo+BHVyG1Q==
-----END CERTIFICATE-----
subject=/C=US/ST=California/L=Mountain View/O=VeriSign, Inc./OU=Production/CN=ww
w.verisign.com
issuer=/C=US/O=RSA Data Security, Inc./OU=Secure Server Certification Authority
---
No client certificate CA names sent
---
SSL handshake has read 1335 bytes and written 302 bytes
---
New, TLSv1/SSLv3, Cipher is DES-CBC3-SHA
Server public key is 1024 bit
SSL-Session:
    Protocol  : SSLv3
    Cipher    : DES-CBC3-SHA
    Session-ID: 2F916C853AD66B5265E7019952E712A175CCB6624AFEAF0F4AF6DECAFFABA57F

    Session-ID-ctx:
    Master-Key: F81FBDB24EA3C86F20C0B53854233FCF2991D3B5692218D53781FB4FEC20B6B6
3AF5799E5999AC9D6E23A6DC07BDCC9A
    Key-Arg   : None
    Start Time: 1008130248
    Timeout   : 7200 (sec)
    Verify return code: 19 (self signed certificate in certificate chain)
---
GET / HTTP/1.0
```

Your screen should be very similar to the prior screen. If not, check your command and make sure you have access to the OpenSSL applications (the OpenSSL directory path is appended to your PATH environment variable) and the files you are point to in the command do in fact exist and you know the password. The interesting thing to do this time is to issue the following command when the connection is made:

GET / HTTP/1.0

This command is presented in the previous screen shot near the bottom of the image. This command is an HTTP request for the selected home page of the SSL host, in this case the Verisign home page should be returned after you press enter twice.


C:\CARoot\cmd.exe

```
e="5" height="102" border="0"></a></td>
      <td width="60%" height="120" valign="middle" align="left">
      &#169; 2001 VeriSign, Inc. All rights reserved. <a href="/legalNotice.html
">Legal&nbsp;Notices</a><br>
      Main Phone: 650-961-7500 &#183; Fax: 650-961-7300<br>
      Sales: 650-426-5115<br>
      Toll Free: 1-866-893-6565<br>
      <br>
      <a href="/truste/index.html"><img src="/images/truste.gif" width="116" hei
ght="31" border="0"></a><br>
      <img src="/images/1ptrans.gif" border="0" height="1" width="1"><br></td>
      <td width="25%" class="copyrt"><a href="javascript:open_window('https://di
gitalid.verisign.com/as2/e40958893f6b2d8139ac2d28747 82dfa')"><img src="/images/s
eals/Secure-White98x102.gif" width="98" height="102" align="right" hspace="10" b
order="0"></a></td>
   </tr>
   </table>
   </td>
</tr>
</table>

</td>
</tr>
</table>

<!-- Begin NewChannel Tags -->
<LAYER NAME="invSpan" VISIBILITY="hidden" TOP="0" LEFT="0"></LAYER>
<NewChannel>
<SCRIPT type='text/javascript' src='http://verisign.webhost.newchannel.com/visit
or/verisign/visitor_tag.cgi'></SCRIPT>
</NewChannel>
<!--  End Newchannel Tags -->

<br><br>

</body>
</html>
closed

C:\CARoot>_
```

You should have received the home page of http://www.verisign.com after issuing the above command. HTTPS can be used with these and other commands to receive secure data from HTTPS servers. This is very convenient for communicating with e-commerce servers. For information on connecting to secure credit card companies you should consider purchasing Visual SSL at http://www.visualSSL.com.

If you would like to separate your key and certificate file you may do the following. First open the localhost.pem file and copy all the data of the certificate section.

```
-----BEGIN CERTIFICATE-----
MIIBzTCCAbqgAwIBAgIQJVgAkqbw4ptI23fbg50KkjAJBgUrDgMCHQUAMGkxEjAQ
BgNVBAMTCWxvY2FsaG9zdDEQMA4GA1UECxMHVGVzdGluZzEPMA0GA1UEChMGVHJp
emVuMREwDwYDVQQHEwhIZWF0aHJvdzEQMA4GA1UECBMHRmxvcmlkYTELMAkGA1UE
BhMCVVMwHhcNMDExMjExMDQxMDUzWhcNMDIxMjExMTAxMDUzWjBpMRIwEAYDVQQD
Ewlsb2NhbGhvc3QxEDAOBgNVBAsTB1Rlc3RpbmcxDzANBgNVBAoTBlRyaXplbjER
MA8GA1UEBxMISGVhdGhyb3cxEDAOBgNVBAgTB0Zsb3JpZGExCzAJBgNVBAYTAlVT
MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCm9xkTPpcMmcNhDvpqpj9zLzPT
qJCL7Yb3qf27teXZxM5lphim8XZ80QxkdJ74UNT+WLapKHD+DXmZ/0UiDUQvO2oV
GLhyD9T3nhowwYXaK2/yegUCfpbpIDefy3AGGj/GbLiL5dIHofOX2DYg8xFOHIo+
EQKnat6RzRuCpo7LtQIDAQABMAkGBSsOAwIdBQADAgC+
-----END CERTIFICATE-----
```

Place the copied text into a new text file in your c:\CARoot\private\IIS5 directory and name it localhost.crt. Next, copy the private key data from the localhost.pem file:

```
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: DES-EDE3-CBC,FC8C9D26EBCD275C

Lt9KdA1IqENQzvo/Y5nmDX2dcQ2tbK4xQiQJKsbqWaUfGtYM/ZC/+zbUsLu9qCIs
mMYZUbOBg0Xc9U5hLeVJNCiotind1ThWSf89Cr8UTpQBBts+kHtzxHNifaaVjU+n
wTf0dg2Jtp0p1Te5Kw5VaGN8OdstCU8KYwWoAxWgD6v7N6P91Zn9DQEz7Z7KR7PD
fWgn2PxmwhHjyTYJtZhkLKT+y/uYV/BRZQfvpbCkP1X09fdzTmHHG32xA/KuSX+C
JsmZs9H8m0rCxWdhY4C0UvmgFmBm4WrCAl2UngoZ3uwvVSCdD79Acuk2u+hC2Hz/
0GY/7k7vbXyOkl9HyTql3XiDA+0yk4PFw/ANXOjB5DAqZ2qKZZzf5jZ8cjEwV6Cw
vD64JHmre7OylQ/ATGyRvM5zQyl8g7uhSoPsnOp73drn7XvpB2gfjtFNNuJ0ckKy
OnDxwAbnSAzjgj3P5USfJujWzsGccVvtWIYxlEe7pd56I7Q2MTC/+GJbSIoZ+jIQ
jO/J9piQ/aWg5tiauTHFXBAiTgEykI4fkcLTucBk10BDyNIYjOF60xL/R2oBaEUF
DAnwqbZ/AscvuuJN2J+nGkrTypMVrCoBGp+QA1ckaNiizgyr4Qn5g99AtrHYyd26
bKhi/NcQFRLLPHYLnD8GG0nNQ3AYpt0MXS6e56LojqIRNN1/sG1wVl2IhDirfZQ1
21zBVx5f9WpWflQesFiBWMJshv7X9WMMvGoKNJNB4eDorf4qAPxEvdcp7/31M7rF
/u896eNQ//ipYAuEU8Ks88c5wyugQPBkcTCtyxudhvtgEddB0/1t0A==
-----END RSA PRIVATE KEY-----
```

Place the copied data into a new text file in your c:\CARoot\private\IIS5 directory and name it localhost.key. You now have two separated files and they each can be used within OpenSSL applications. Remember though that you have separated your public certificate and private key file and as such you must specify both for any OpenSSL applications that require a certificate and key file.

We have digressed from the original format slightly to demonstrate some of the ways OpenSSL can be utilized in Windows operating systems. At this point we will demonstrate the more utilized options with the PKCS12 utility.

-in [filename]
Use this option to set the PKCS#12 certificate to be processed. The following command will parse the localhost.pfx and place its private key and public certificate into out.pem.

```
C:\CARoot>pkcs12 –in localhost.pfx –out out.pem
```

-out [filename]
Use this option to specify the output file that will hold the key and certificate in one file.

```
C:\CARoot>pkcs12 –in c:\CARoot\private\IIS5\localhost.pfx –out out.pem
```

-nokeys
Use this option to output only a certificate. You will only be prompted for a password to the PKCS#12 file (if encrypted with a password). If you want to use the newly created certificate in windows you must remove the extra header information from the –out file after it is created.

```
C:\CARoot>pkcs12 –in localhost.pfx –nokeys –out out.pem
```

-info
This options really only add a little information about the process that is taking place with the command.

```
C:\CARoot>pkcs12 –in localhost.pfx –info –out out.pem
```

-des, -des3, -idea
Use one of these options to specify the encryption algorithm to protect the key.

```
C:\CARoot>pkcs12 –in localhost.pfx –des3 –out.pem
```

-nodes
Use this option to NOT encrypt the keys. This means the keys will not be password protected for use. Not recommended.

```
C:\CARoot>pkcs12 –in localhost.pfx –nodes –out.pem
```

-noout
Select this option to keep the utility from creating any output. Usually used with –info switch. The following example demonstrates the use of sending information about a certificate contained in the directory c:\CARoot\private\IIS to the screen.

```
C:\CARoot>pkcs12 –in private\IIS\localhost.pfx –noout –info
```

-password [pass:password]
Use this option when specifying a password that must be used to access a file when creating a PKCS#12 file. This format will have the password displayed on

the command line. For example, if your password was visualssl then the format would be:

```
C:\CARoot>pkcs12 –export –in localhost.pem –out mypkcs12.pfx
        –password pass:visualssl
```

-inkey [filename]
If using this option then the –in file will usually be just a certificate. If both the certificate and key file are in the same file, then just use –in file.

```
C:\CARoot>pkcs12 –export –in private\temp.crt –inkey private\temp.key –out
        mypkcs12.pfx
```

-export
Use this option to CREATE a PKCS#12 file. You must supply at least one private key and one certificate usually in one file. This next example will take a .pfx file and parse it into a .pem file which will have the key and certificate contained within it encrypted.

```
C:\CARoot>pkcs12 –in localhost.pfx –out localhost.pem
```

The next line will then take this localhost.pem file and turn it back into a PKCS#12 file using the password that was used:

```
C:\CARoot>pkcs12 –export –in localhost.pem –out mypkcs12.pfx
        –password pass:mypass
```

At this point the file can be double clicked on in Microsoft Windows and imported into a Trusted Certificate Store. The next example shows how to use an already created certificate file and a key file to generate a PKCS12 file.

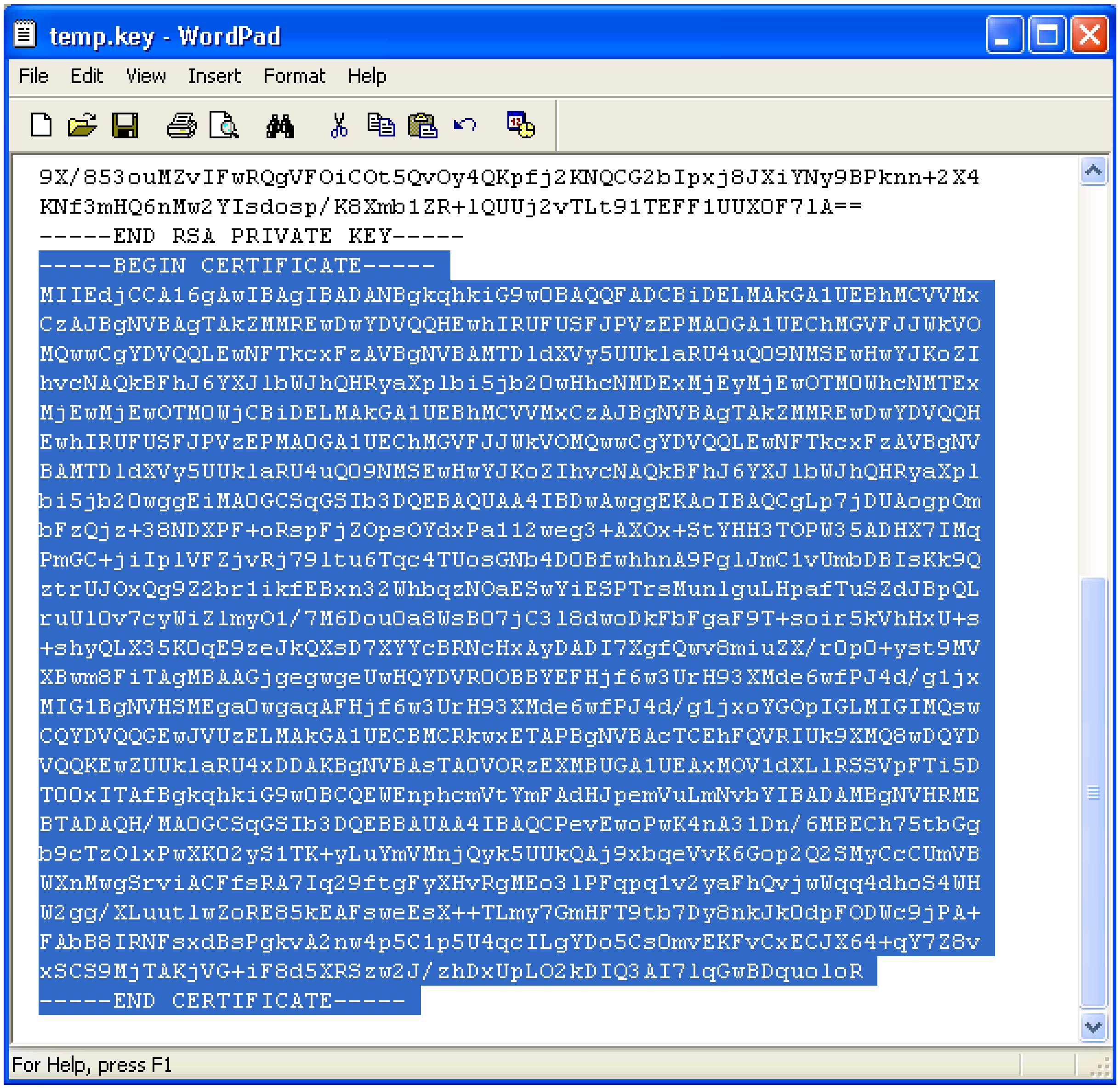
Step 1) Generate a Key File

C:\CARoot>*genrsa –out private\temp.key –rand private\.rnd –des3 2048*

Step 2) Generate a Certificate File from the above key

C:\CARoot>*req –new –x509 –days 3650 –key private\temp.key –out private\temp.crt –config openssl.cnf*

Step 3) Copy the Certificate File data into the temp.key file

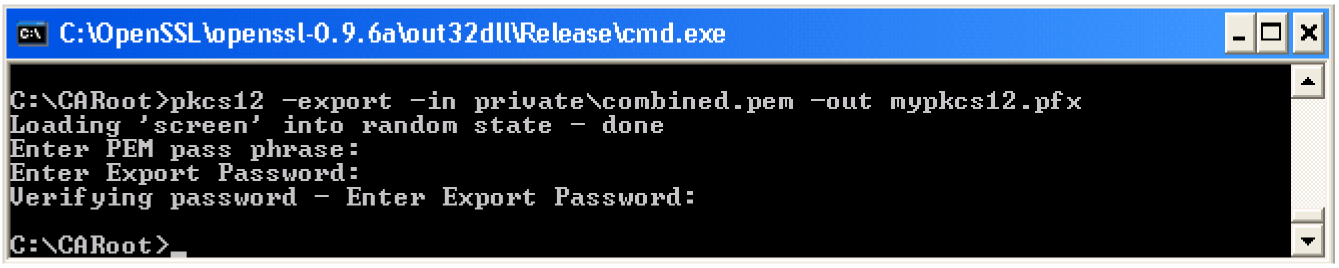


```
9X/853ouMZvIFwRQgVFOiCOt5QvOy4QKpfj2KNQCG2bIpxj8JXiYNy9BPknn+2X4
KNf3mHQ6nMw2YIsdosp/K8Xmb1ZR+1QUUj2vTLt91TEFF1UUXOF71A==
-----END RSA PRIVATE KEY-----
-----BEGIN CERTIFICATE-----
MIIEdjCCA16gAwIBAgIBADANBgkqhkiG9w0BAQQFADCBiDELMAkGA1UEBhMCVVMx
CzAJBgNVBAgTAkZMMREwDwYDVQQHEwhIRUFUSFJPVzEPMA0GA1UEChMGVFJJWkVO
MQwwCgYDVQQLEwNFTkcxFzAVBgNVBAMTD1dXVy5UUklaRU4uQ09NMSEwHwYJKoZI
hvcNAQkBFhJ6YXJlbWJhQHRyaXplbi5jb20wHhcNMDExMjEyMjEwOTM0WhcNMTEx
MjEwMjEwOTM0WjCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAkZMMREwDwYDVQQH
EwhIRUFUSFJPVzEPMA0GA1UEChMGVFJJWkVOMQwwCgYDVQQLEwNFTkcxFzAVBgNV
BAMTD1dXVy5UUklaRU4uQ09NMSEwHwYJKoZIhvcNAQkBFhJ6YXJlbWJhQHRyaXpl
bi5jb20wggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQCgLp7jDUAogpOm
bFzQjz+38NDXPF+oRspFjZOpsOYdxPa112weg3+AXOx+StYHH3TOPW35ADHX7IMq
PmGC+jiIplVFZjvRj79ltu6Tqc4TUosGNb4DOBfwhhnA9PglJmC1vUmbDBIsKk9Q
ztrUJOxQg9Z2br1ikfEBxn32WhbqzNOaESwYiESPTrsMunlguLHpafTuSZdJBpQL
ruUlOv7cyWiZlmyO1/7M6DouOa8WsBO7jC3l8dwoDkFbFgaF9T+soir5kVhHxU+s
+shyQLX35KOqE9zeJkQXsD7XYYcBRNcHxAyDADI7XgfQwv8miuZX/rOpO+yst9MV
XBwm8FiTAgMBAAGjgegwgeUwHQYDVROOBBYEFHjf6w3UrH93XMde6wfPJ4d/g1jx
MIG1BgNVHSMEga0wgaqAFHjf6w3UrH93XMde6wfPJ4d/g1jxoYGOpIGLMIGIMQsw
CQYDVQQGEwJVUzELMAkGA1UECBMCRkwxETAPBgNVBAcTCEhFQVRIUk9XMQ8wDQYD
VQQKEwZUUklaRU4xDDAKBgNVBAsTA0VORzEXMBUGA1UEAxMOV1dXLlRSSVpFTi5D
TOOxITAfBgkqhkiG9w0BCQEWEnphcmVtYmFAdHJpemVuLmNvbYIBADAMBgNVHRME
BTADAQH/MA0GCSqGSIb3DQEBBAUAA4IBAQCPevEwoPwK4nA31Dn/6MBECh75tbGg
b9cTzOlxPwXKO2yS1TK+yLuYmVMnjQyk5UUkQAj9xbqeVvK6Gop2Q2SMyCcCUmVB
WXnMwgSrviACFfsRA7Iq29ftgFyXHvRgMEo31PFqpq1v2yaFhQvjwWqq4dhoS4WH
W2gg/XLuutlwZoRE85kEAFsweEsX++TLmy7GmHFT9tb7Dy8nkJkOdpFODWc9jPA+
FAbB8IRNFsxdBsPgkvA2nw4p5C1p5U4qcILgYDo5CsOmvEKFvCxECJX64+qY7Z8v
xSCS9MjTAKjVG+iF8d5XRSzw2J/zhDxUpLO2kDIQ3AI7lqGwBDquoloR
-----END CERTIFICATE-----
```

As you can see from the above screen shot we have pasted the temp.crt information (by opening temp.crt in WordPad) into the temp.key file. Make sure there are no lines between the dashed lines and no extra lines at the bottom.

<u>Step 4) Save the file as combined.pem</u>

<u>Step 5) Create the PKCS#12 file</u>



```
C:\CARoot>pkcs12 -export -in private\combined.pem -out mypkcs12.pfx
Loading 'screen' into random state - done
Enter PEM pass phrase:
Enter Export Password:
Verifying password - Enter Export Password:

C:\CARoot>_
```

From the above MS-DOS command prompt screen shot we have issued the following command:

C:\CARoot>pkcs12 –export –in private\combined.pem –out mypkcs12.pfx

If you now double click on the new mypkcs12.pfx file in the C:\CARoot directory you will be prompted by windows to import the file. You can just press Cancel if you wish or go ahead and import the file. If you did not get the desired results make sure you have followed the instructions carefully realizing that creating key files and certificates were covered in great detail early in this manual and are sometimes very tricky.

Ok, here is an easier way if you don't want to copy text from one file to another by simply using the generated Certificate temp.crt and the Key file temp.key.

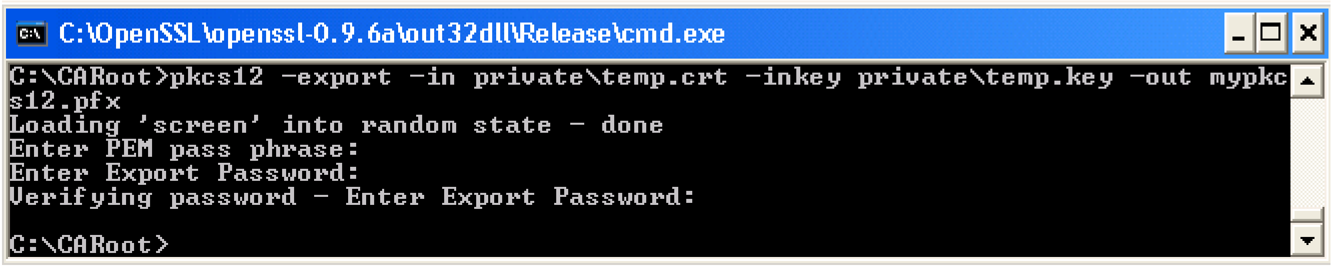
Step 1) Generate a Key File

C:\CARoot>*genrsa –out private\temp.key –rand private\.rnd –des3 2048*

Step 2) Generate a Certificate File from the above key

C:\CARoot>*req –new –x509 –days 3650 –key private\temp.key –out private\temp.crt –config openssl.cnf*

Step 3) Create a PKCS#12 File from the temp.key and temp.crt



```
C:\CARoot>pkcs12 -export -in private\temp.crt -inkey private\temp.key -out mypkc
s12.pfx
Loading 'screen' into random state - done
Enter PEM pass phrase:
Enter Export Password:
Verifying password - Enter Export Password:

C:\CARoot>
```

The command is virtually the same as before except we are explicitly telling the PKCS utility that there is a separate key file through the use of the –inkey switch. The utility would then expect the –in file to be just a certificate:

C:\CARoot>pkcs12 –export –in private\temp.crt –inkey private\temp.key –out mypkcs12.pfx

Authenticode and Digitally Signing Your Applications

For those that are interested in learning how to create ActiveX controls and have them operate from within Internet Explorer even when the security is High, then you should sign your ActiveX controls and applications with an Authenticode Certificate which is nothing more than a PKCS#12 signed certificate.

The process is relatively straight forward and summarized as follows:

1) Create a Certificate Request from Internet Information Server (described in PKCS#12 section)
2) Sign the Certificate as follows:

```
C:\CARoot>ca –in private\IIS5\certreq.txt –key private\ca.key –out
        private\IIS\trizen.cer –policy policy_anything –config openssl.cnf
```

3) Complete the Internet Information Server certificate process by returning to the same place in IIS to create the certificate request. Import the newly signed certificate.
4) Use Microsoft Management Console to export the certificate as a PKCS#12 certificate to c:\CARoot\private\IIS5\trizen.pfx
5) Parse the newly created trizen.pfx file using the PKCS12 utility:

```
C:\CARoot>pkcs12 –in private\IIS5\trizen.pfx –out private\IIS5\trizen.pem
```

6) Separate the key portion from the newly created trizen.pem file making it compatible to Windows (removing everything but the ----BEGIN RSA PRIVATE KEY ---- and ---- END RSA PRIVATE KEY -----.
7) Save the newly created key file to c:\CARoot\private\IIS5\trizen.key
8) Create a PKCS#12 certificate file using the signed certificate trizen.crt and the exported key file trizen.key using the following command:

```
C:\CARoot\>pkcs12 –export –in private\IIS5\trizen.cer –inkey
        private\IIS5\trizen.key –out private\IIS5\newtrizen.pfx
```

9) Double-click on the newtrizen.pfx file and import it into windows.
10) Download the Authenticode SDK from Microsoft.com and run the signcode.exe application.
11) Select your ActiveX control or .cab file to sign and use the certificate store. It should have the newly imported PKCS#12 certificate.
12) Complete the code signing process, realizing that you should let Microsoft select the Security Provider which should be captured already in the operating system during the import.

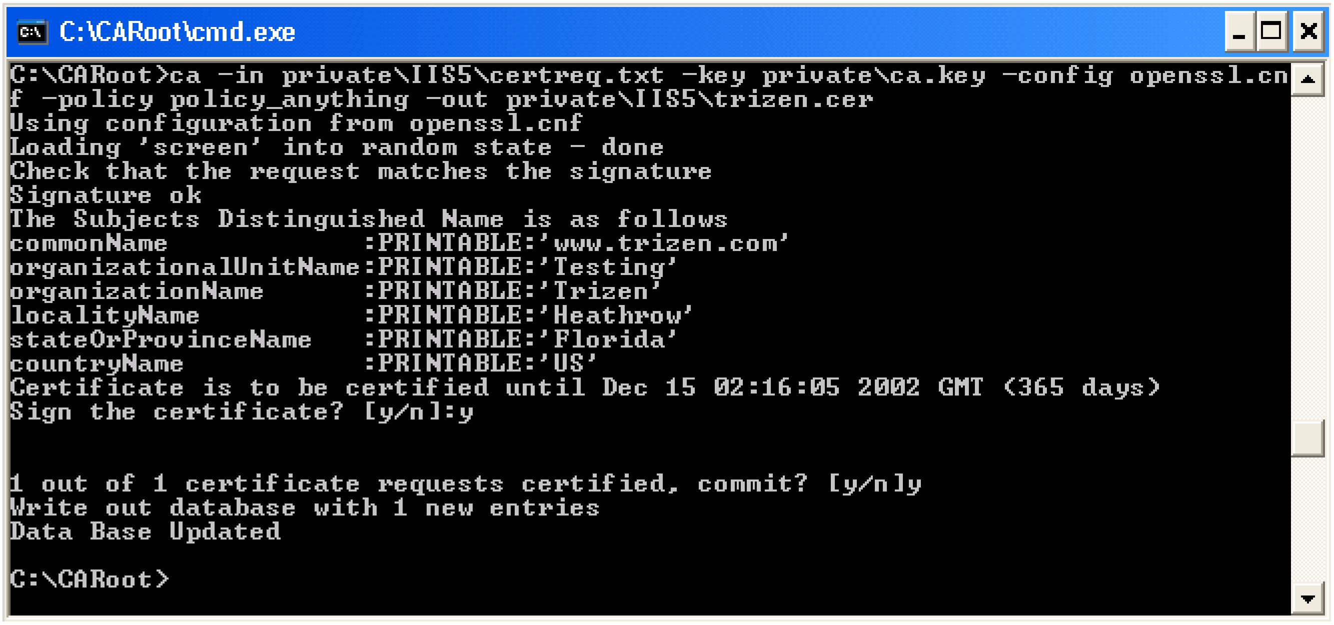
Did you get all that? If not here is the process in more excruciating detail. Remember, if you're interested in learning about Authenticode, you should review the Microsoft technologies of Cabinet files, and ActiveX distribution. For more information on ActiveX distribution, safe code, digitally signed code, you should consider purchasing our VisualSSL product.

Complete Detail of Creating Authenticode Applications

Create a Certificate Request using the methods described in the **Creating PKCS#12 Certificates in Internet Information Server** section. Make sure you save this file in c:\CARoot\private\IIS5\certreq.txt.

Sign the certificate request using the following command:

```
C:\CARoot>ca –in private\IIS5\certreq.txt –key private\ca.key –config
      openssl.cnf –policy policy_anything –out private\IIS5\trizen.cer
```

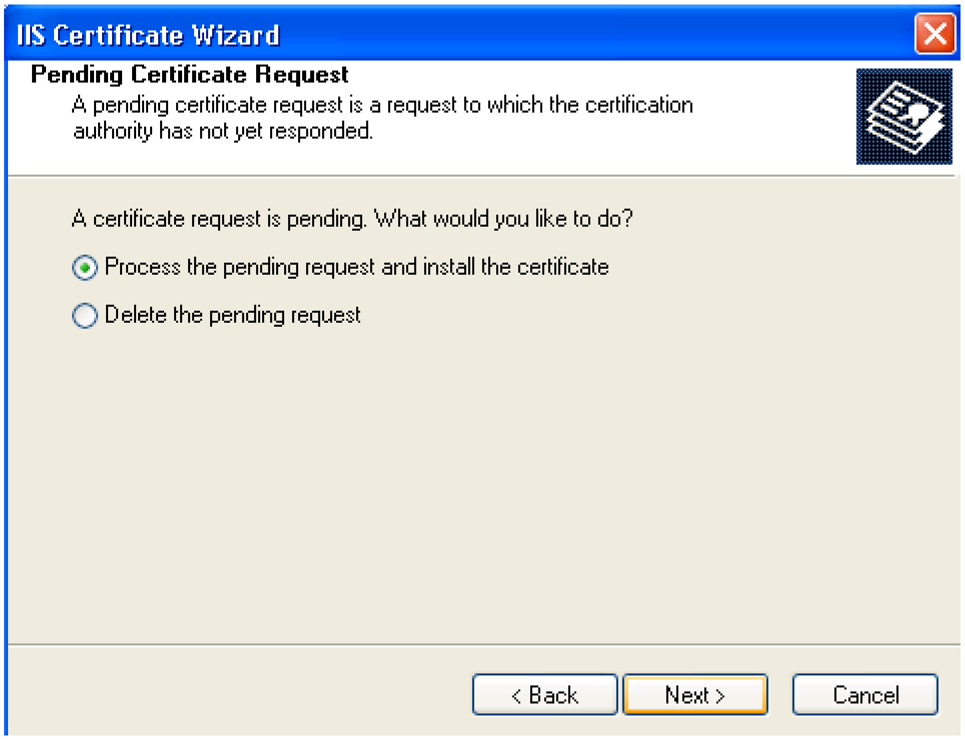


```
C:\CARoot\cmd.exe
C:\CARoot>ca -in private\IIS5\certreq.txt -key private\ca.key -config openssl.cn
f -policy policy_anything -out private\IIS5\trizen.cer
Using configuration from openssl.cnf
Loading 'screen' into random state - done
Check that the request matches the signature
Signature ok
The Subjects Distinguished Name is as follows
commonName            :PRINTABLE:'www.trizen.com'
organizationalUnitName:PRINTABLE:'Testing'
organizationName      :PRINTABLE:'Trizen'
localityName          :PRINTABLE:'Heathrow'
stateOrProvinceName   :PRINTABLE:'Florida'
countryName           :PRINTABLE:'US'
Certificate is to be certified until Dec 15 02:16:05 2002 GMT (365 days)
Sign the certificate? [y/n]:y


1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated

C:\CARoot>
```

You must make sure that you have setup your Certificate Authority structure and that your paths are accurate including your PATH environment variable which must include the directory to the OpenSSL applications.

Once the certificate has been signed, you must make sure that you modify the certificate to conform to the Windows Operating System. See Adjusting the Certificates for Windows Operating Systems for more information.

At this point you must return to the Internet Information Server security area and complete the Certificate Request Process. Return to the same location where you first initiated the Certificate Request. Upon returning to this dialog you should see the following:



Pending Certificate Request

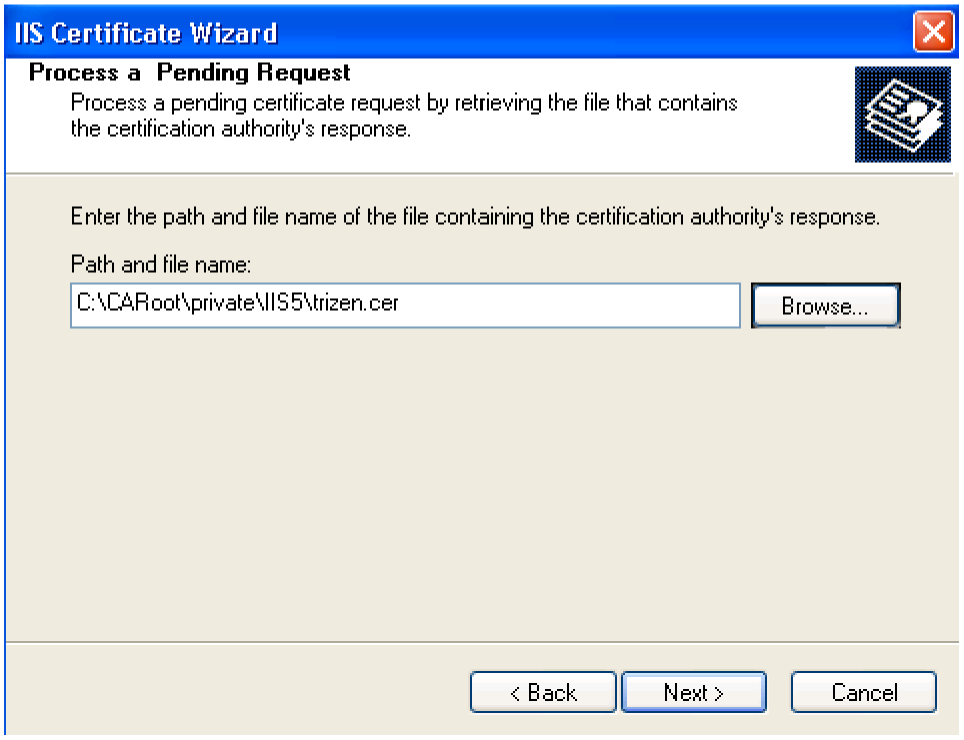
A pending certificate request is a request to which the certification authority has not yet responded.

A certificate request is pending. What would you like to do?

[✓ Process the pending request and install the certificate]

Delete the pending request

Notice that IIS is waiting for the signed certificate that was created earlier. At this point press the Next > button and navigate to the trizen.crt file.



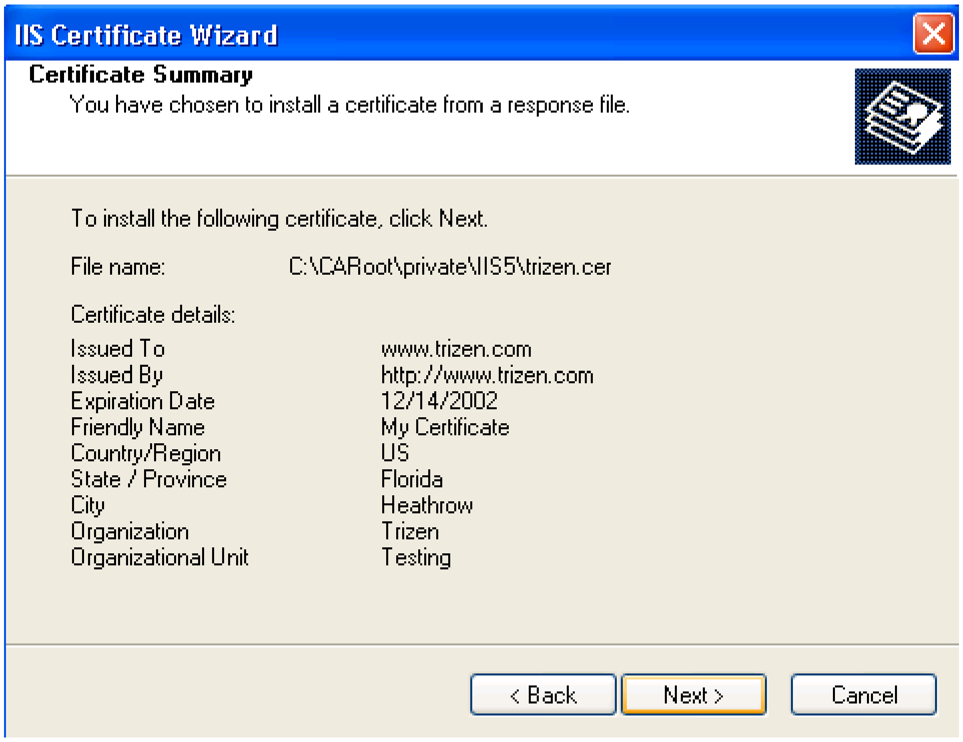
Process a Pending Request

Process a pending certificate request by retrieving the file that contains the certification authority's response.

Enter the path and file name of the file containing the certification authority's response.

Path and file name: C:\CARoot\private\IIS5\trizen.cer

At this point press the Next > button.



Certificate Summary

You have chosen to install a certificate from a response file.

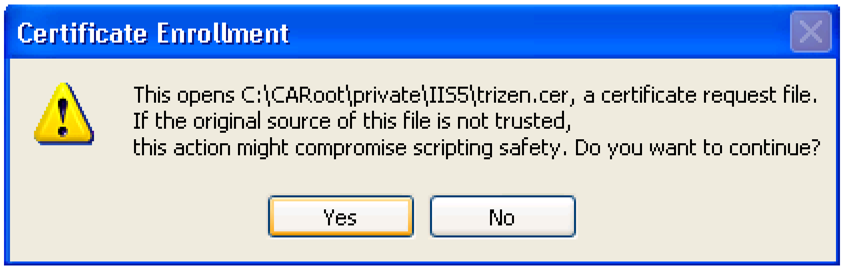
To install the following certificate, click Next.

File name: C:\CARoot\private\IIS5\trizen.cer

Certificate details:

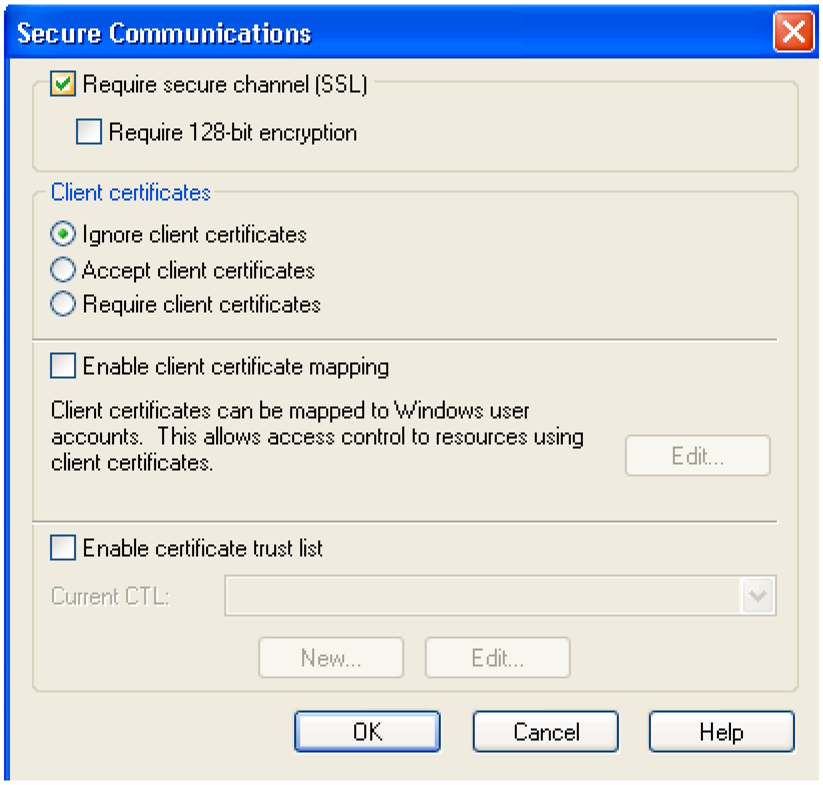
Issued To	www.trizen.com
Issued By	http://www.trizen.com
Expiration Date	12/14/2002
Friendly Name	My Certificate
Country/Region	US
State / Province	Florida
City	Heathrow
Organization	Trizen
Organizational Unit	Testing

If you do not see the above dialog after pressing the Next > button review your certificate by double clicking on it making sure Windows recognizes the certificate. After pressing the Next > button you will likely get the following:



This opens C:\CARoot\private\IIS5\trizen.cer, a certificate request file.
If the original source of this file is not trusted,
this action might compromise scripting safety. Do you want to continue?

Just press Yes to continue. Windows will notify you that the process is complete. You may have noticed that at this point you are performing the same actions to create an SSL server. In fact press the Edit button in the secure communications area of the Directory Security Tab of the Default Web Site properties. You should see the following dialog which will allow you to make your server SSL enabled:


Secure Communications

[✓] Require secure channel (SSL)

[ ] Require 128-bit encryption

Client certificates

(●) Ignore client certificates

( ) Accept client certificates

( ) Require client certificates

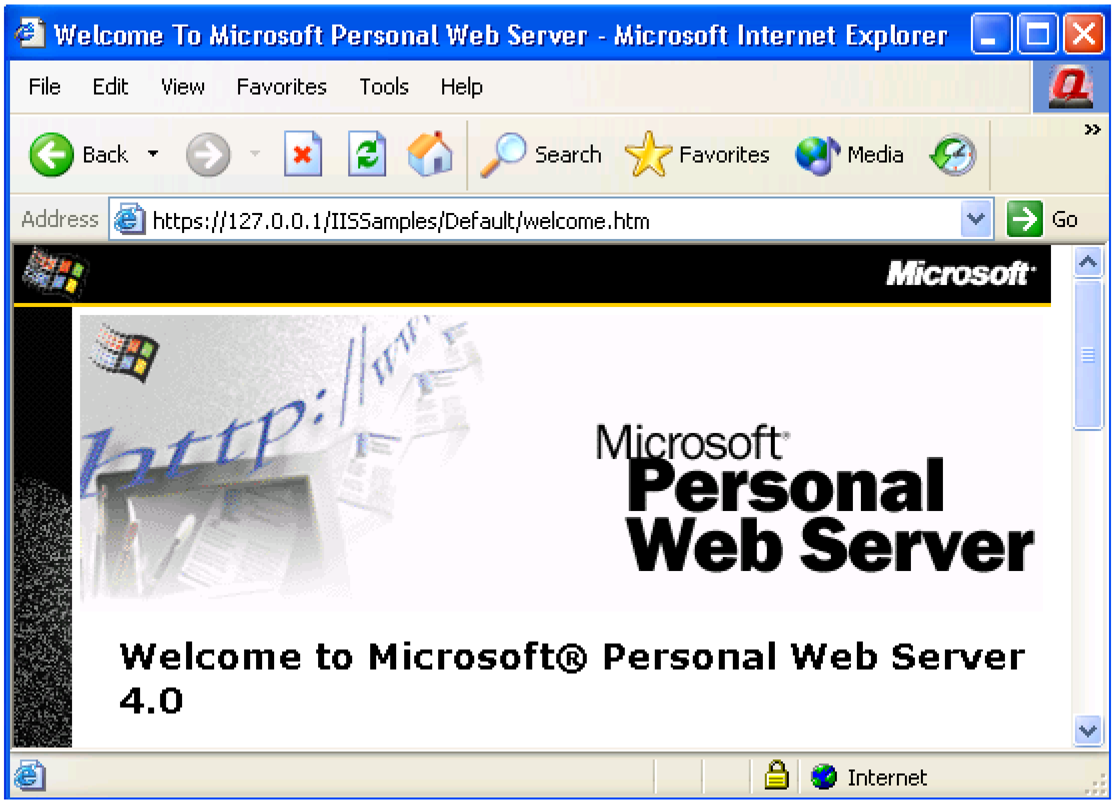
[ ] Enable client certificate mapping

Client certificates can be mapped to Windows user accounts. This allows access control to resources using client certificates.

[ ] Enable certificate trust list

Current CTL:

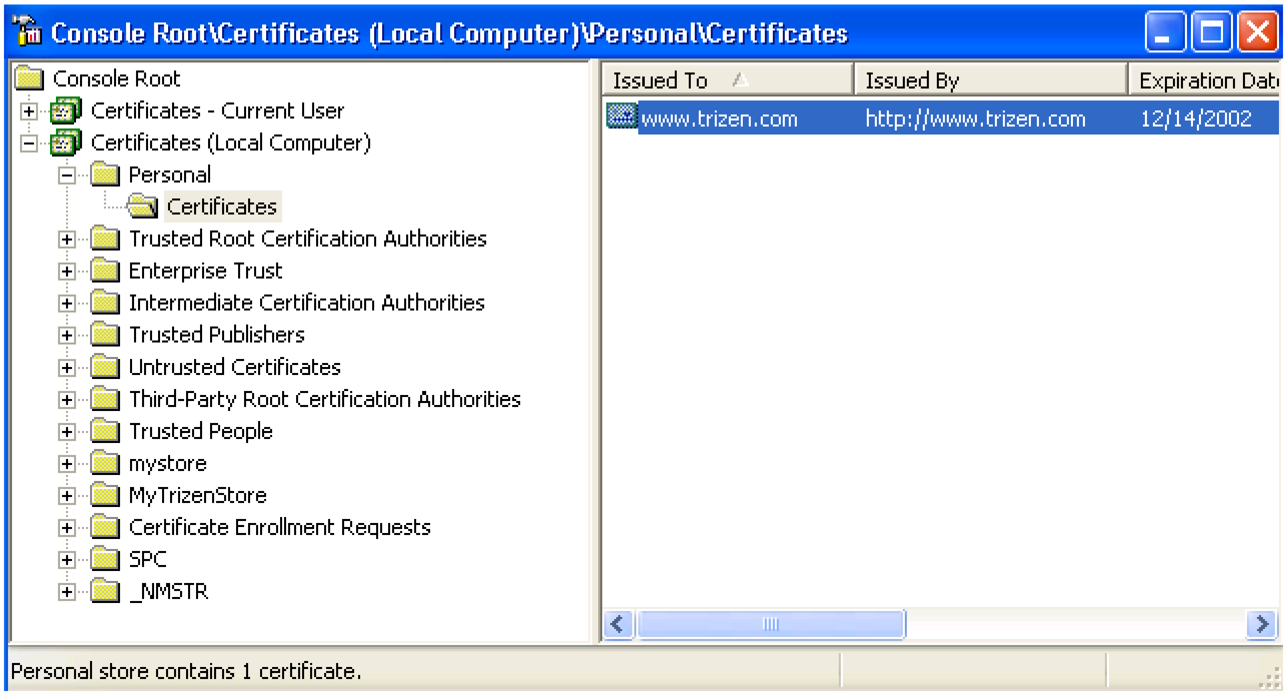
If you select the "Require secure channel (SSL)" and press OK, your server will only accept secure connections! To see this in action after you press the OK button, open up Internet Explorer and type in https://127.0.0.1 and you should see your default web page, but this time the secure connection icon can be found in your web browser's status bar.


Welcome To Microsoft Personal Web Server - Microsoft Internet Explorer
File
Edit
View
Favorites
Tools
Help
Back
Search
Favorites
Media
Address
https://127.0.0.1/IISSamples/Default/welcome.htm
Go
Microsoft
http://
Microsoft Personal Web Server
Internet

Welcome to Microsoft® Personal Web Server 4.0

You must realize that since Internet browsers have a limited list of trusted certificates, others may not be able to connect to your site because their browsers do not know of your certificate, they will likely be prompted to continue loading the page.

Now back to Authenticode. After you have completed the CSR process by first creating a certificate request, signing the request, and then importing the signed certificate into IIS, you must now export the key and certificate as a PKCS#12 file. This is the same process as described in the PKCS12 Utility application and will not be repeated here, however, when you export the PKCS#12 certificate, make sure you export it to c:\CARoot\private\IIS5\trizen.pfx. The store that it will likely be in will be in the Local Computer account in the Personal store, you may have to search to find your certificate, looking for the certificate property such as http://www.trizen.com which is usually the common name of the server.


Console Root\Certificates (Local Computer)\Personal\Certificates
Console Root
Certificates - Current User
Certificates (Local Computer)
Personal
Certificates
Trusted Root Certification Authorities
Enterprise Trust
Intermediate Certification Authorities
Trusted Publishers
Untrusted Certificates
Third-Party Root Certification Authorities
Trusted People
mystore
MyTrizenStore
Certificate Enrollment Requests
SPC
_NMSTR

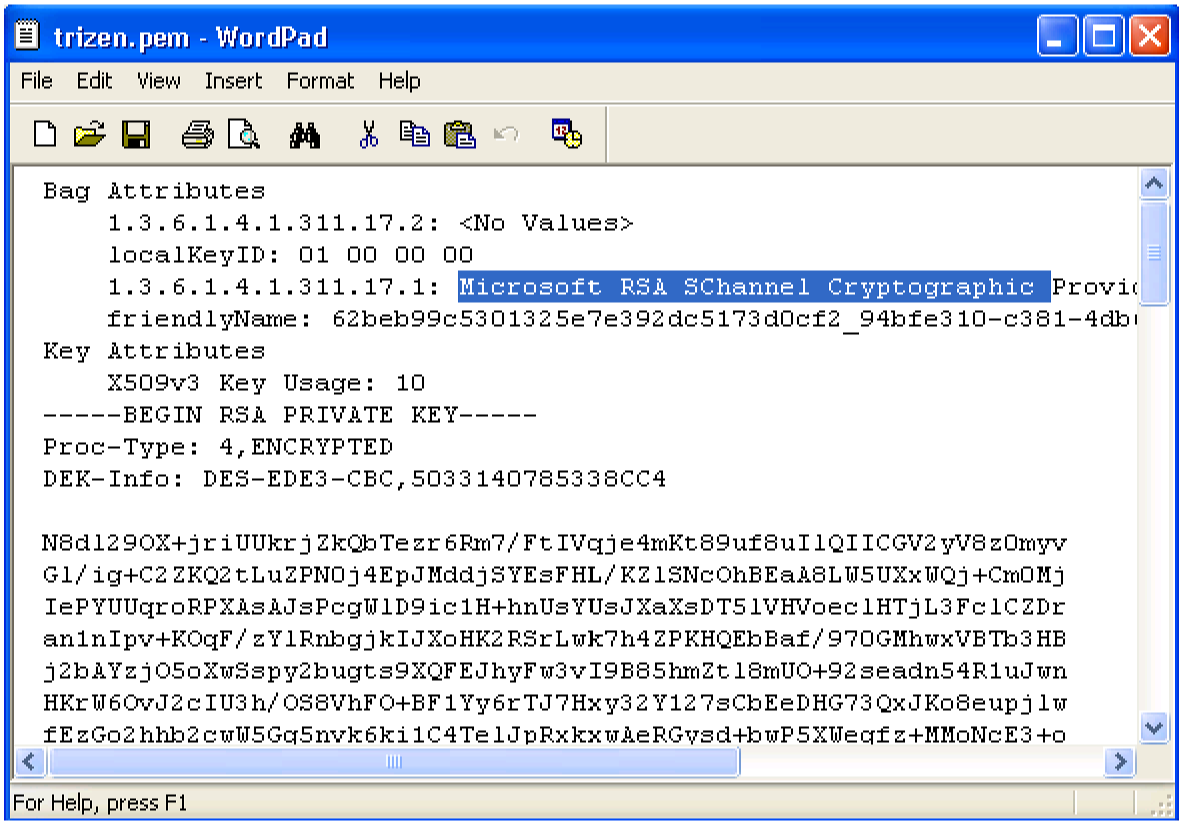
Issued To	Issued By	Expiration Date
www.trizen.com	http://www.trizen.com	12/14/2002

Personal store contains 1 certificate.

After you successfully exported the Key and Certificates (by right mouse clicking on the certificate and selecting export) into a PKCS#12 file (trizen.pfx) you must now parse the file so that you can have an independent key file by first issuing the following command:

```
C:\CARoot>pkcs12 –in private\IIS5\trizen.pfx –out private\IIS5\trizen.pem
```

The above command will export the key associated with the certificate that was created along with the certificate request. We had to export it in this format to get it out of the Windows operating system. Your file should look similar to:



```
Bag Attributes
    1.3.6.1.4.1.311.17.2: <No Values>
    localKeyID: 01 00 00 00
    1.3.6.1.4.1.311.17.1: Microsoft RSA SChannel Cryptographic Provi
    friendlyName: 62beb99c5301325e7e392dc5173d0cf2_94bfe310-c381-4db
Key Attributes
    X509v3 Key Usage: 10
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: DES-EDE3-CBC,5033140785338CC4

N8dl29OX+jriUUkrjZkQbTezr6Rm7/FtIVqje4mKt89uf8uIlQIICGV2yV8z0myv
Gl/ig+C2ZKQ2tLuZPN0j4EpJMddjSYEsFHL/KZlSNcOhBEaA8LW5UXxWQj+Cm0Mj
IePYUUqroRPXAsAJsPcgWlD9ic1H+hnUsYUsJXaXsDT5lVHVoeclHTjL3FclCZDr
an1nIpv+KOqF/zYlRnbgjkIJXoHK2RSrLwk7h4ZPKHQEbBaf/970GMhwxVBTb3HB
j2bAYzjO5oXwSspy2bugts9XQFEJhyFw3vI9B85hmZtl8mUO+92seadn54R1uJwn
HKrW6OvJ2cIU3h/OS8VhFO+BF1Yy6rTJ7Hxy32Y127sCbEeDHG73QxJKo8eupjlw
fEzGo2hhb2cwW5Gg5nvk6ki1C4TelJpRxkxwAeRGysd+bwP5XWegfz+MMoNcE3+o
```

Notice that in the above file the property of Microsoft RSA Schannel Cryptographic Provider. This can only be accomplished by creating a certificate from within IIS and selecting this option during the creation phase.

At this point you must separate the private key from the rest of the file by following the similar approach to separating certificates manually from the extraneous information. Your key should look like the following:

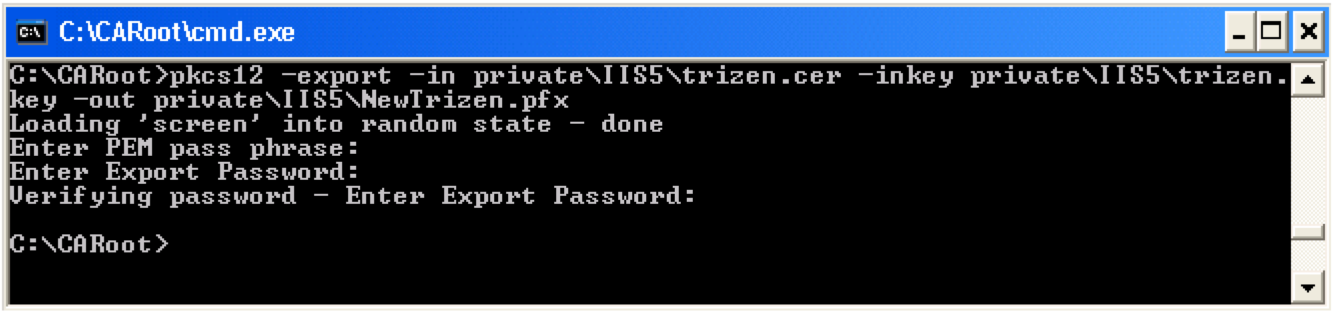
```
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: DES-EDE3-CBC,5033140785338CC4

N8dl29OX+jriUUkrjZkQbTezr6Rm7/FtIVqje4mKt89uf8uIlQIICGV2yV8z0myv
Gl/ig+C2ZKQ2tLuZPN0j4EpJMddjSYEsFHL/KZlSNcOhBEaA8LW5UXxWQj+Cm0Mj
IePYUUqroRPXAsAJsPcgWlD9ic1H+hnUsYUsJXaXsDT5lVHVoeclHTjL3FclCZDr
an1nIpv+KOqF/zYlRnbgjkIJXoHK2RSrLwk7h4ZPKHQEbBaf/970GMhwxVBTb3HB
j2bAYzjO5oXwSspy2bugts9XQFEJhyFw3vI9B85hmZtl8mUO+92seadn54R1uJwn
HKrW6OvJ2cIU3h/OS8VhFO+BF1Yy6rTJ7Hxy32Y127sCbEeDHG73QxJKo8eupjlw
fEzGo2hhb2cwW5Gg5nvk6ki1C4TelJpRxkxwAeRGysd+bwP5XWegfz+MMoNcE3+o
3a90E0Qi/vjp/fkbrjpwn62DSIwEypi7CkStFfrGxkWsV7HPkQUrnZWFCptE0w8b
0qoI1D3EslvC3a/e4XotrpVnQQAiE+Ljln7/IN6k+/96TxKTGeFK/RnCRsiPwTvr
rNEzbrWtS2guxFhDWrJHoNBJM8qW3sp+Gd1MW4L2jTEMSZO0pjCtU9wsGt2uPh+v
YU2Y2iBwlDCbDFlM17yqZdniVduExrR80HkmEBi4gzp8FmUwNu8J2Gm/bhEi5D8n
H+GXTVaagcEx2HfofdwM5PoXwLocugIGKlXjxo9wZA9OxqYoUDmE64+PMMhFmFJF
SMCWUZixmtxTB8obRr5fcePSzgOYJTHnejqPD/pHC9U=
-----END RSA PRIVATE KEY-----
```

Save this key information to c:\CARoot\private\IIS5\trizen.key. We now have several files that will be utilized:

File	Description
Certreq.txt	Our IIS Certificate Request
Trizen.cer	The Signed Certificate by our CA
Trizen.key	The Exported key of the Certificate Request. From Trizen.pem
Trizen.pfx	The Exported PKCS#12
Trizen.pem	The Parsed file of the PKCS#12 file. Certificate and Key.

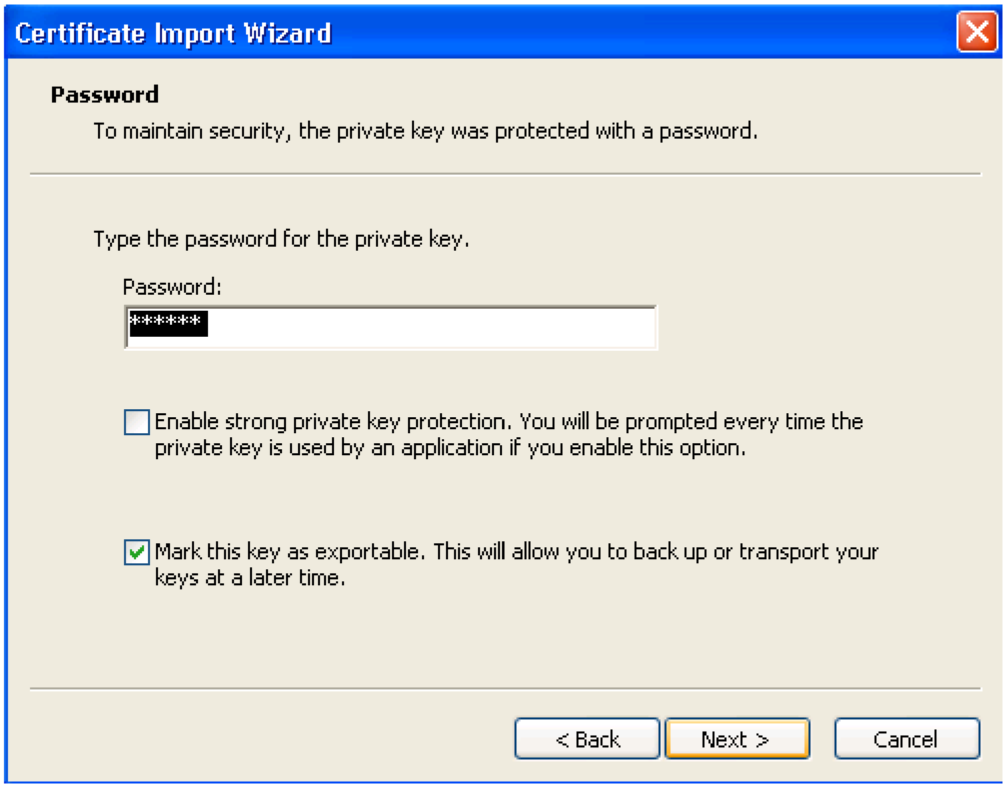
We must now build a PKCS#12 file which will be imported into Windows. Type the following command to create a new PKCS#12 certificate:

```
C:\CARoot>pkcs12 –export –in private\IIS5\trizen.cer –inkey private\IIS5\trizen.key
        -out private\IIS5\NewTrizen.pfx
```



```
C:\CARoot>pkcs12 -export -in private\IIS5\trizen.cer -inkey private\IIS5\trizen.
key -out private\IIS5\NewTrizen.pfx
Loading 'screen' into random state - done
Enter PEM pass phrase:
Enter Export Password:
Verifying password - Enter Export Password:

C:\CARoot>
```

This PKCS#12 file must now be imported into windows. Simply double click on the newly created NewTrizen.pfx file. Windows will prompt you to begin the Certificate Import Wizard. Just continue until you get to the following dialog:


Certificate Import Wizard

Password

To maintain security, the private key was protected with a password.

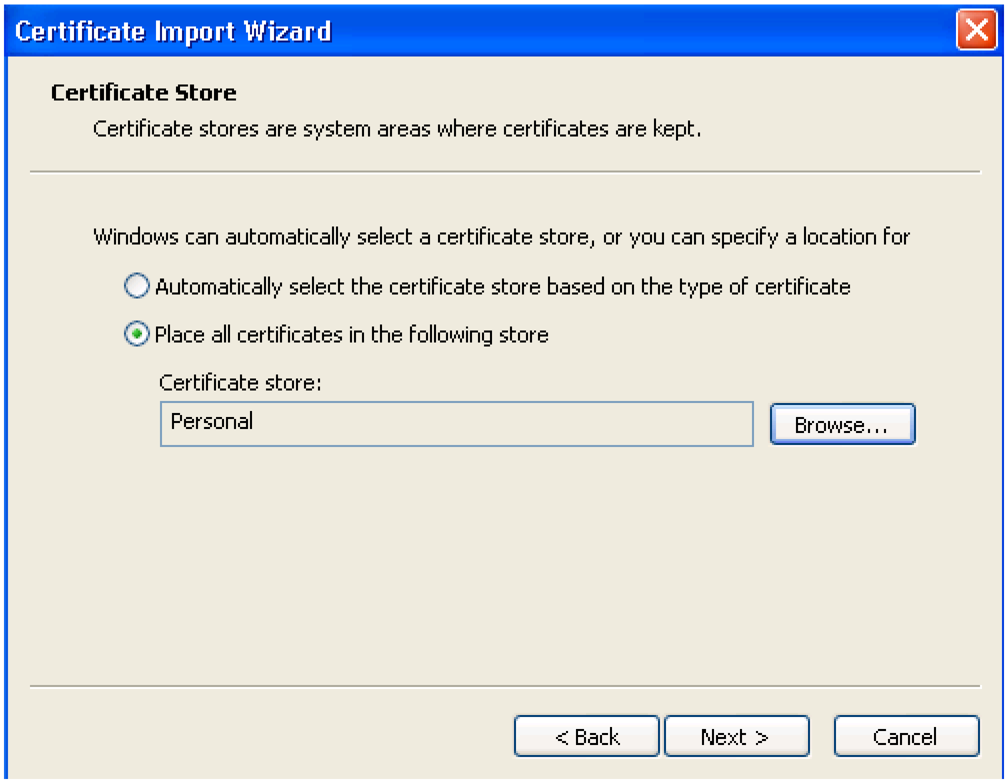
Type the password for the private key.

Password: *******

☐ Enable strong private key protection. You will be prompted every time the private key is used by an application if you enable this option.

☑ Mark this key as exportable. This will allow you to back up or transport your keys at a later time.

Type in the password for your private key, this was set when you issued your certificate request. Also, check the "Mark this key as exportable. This will allow you to back up or transport your keys at a later time." Press Next > to continue the import process:


Certificate Import Wizard

Certificate Store

Certificate stores are system areas where certificates are kept.

Windows can automatically select a certificate store, or you can specify a location for

○ Automatically select the certificate store based on the type of certificate

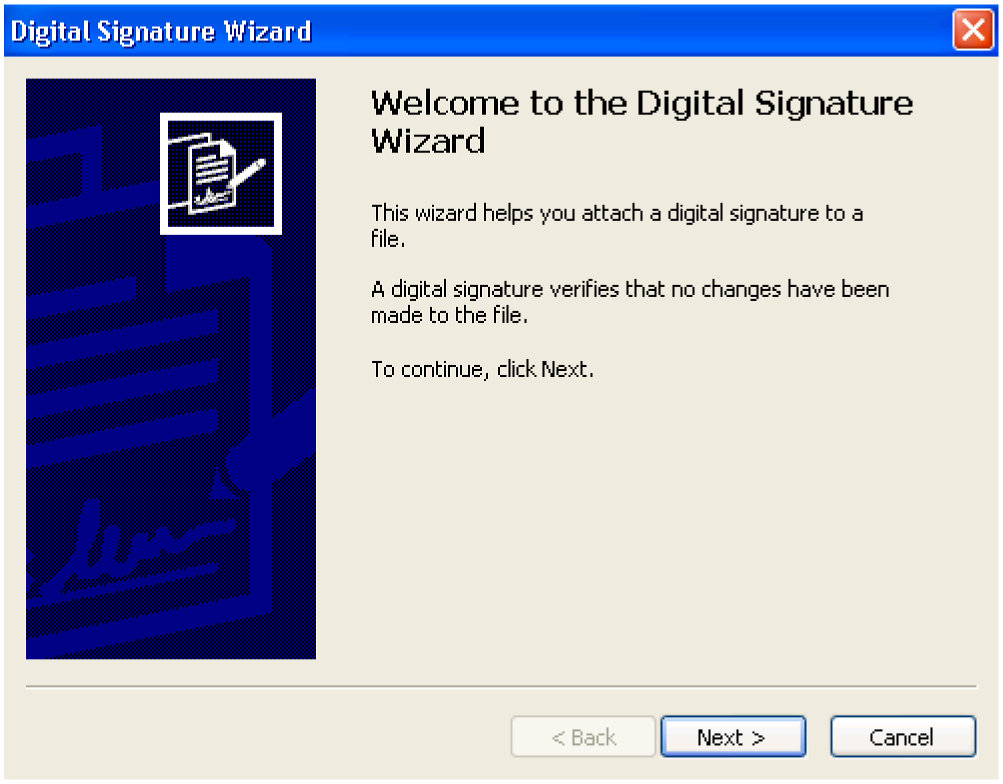
● Place all certificates in the following store

Certificate store: Personal

Make sure you place the certificate into the Personal Store by browsing for its location. Press Next > to continue the process. The next dialog will finish the process. Just press Finish, or Ok to complete the import.

You now have all the items you need to successfully add a digital signature to your applications. At this point you should have already created your ActiveX controls and either packed them into a Microsoft Cabinet file or left them as an .ocx ready for distribution over the internet. In order to completely utilize your application in Internet Explorer without those irritating dialogs, you must mark your application as “safe” using and you must use Authenticode. Both of which still don’t guarantee safety, but do give your customers a better feeling.

If you have not already done so, download Microsoft’s Authenticode SDK at http://msdn.microsoft.com/downloads/ where you should look for Authenticode. Download your version. Once you have the SDK installed you need to run the signcode.exe application only after you have built your ActiveX control or Cabinet file. For information on building a successful Internet Explorer ActiveX control, you should purchase VisualSSL for information.



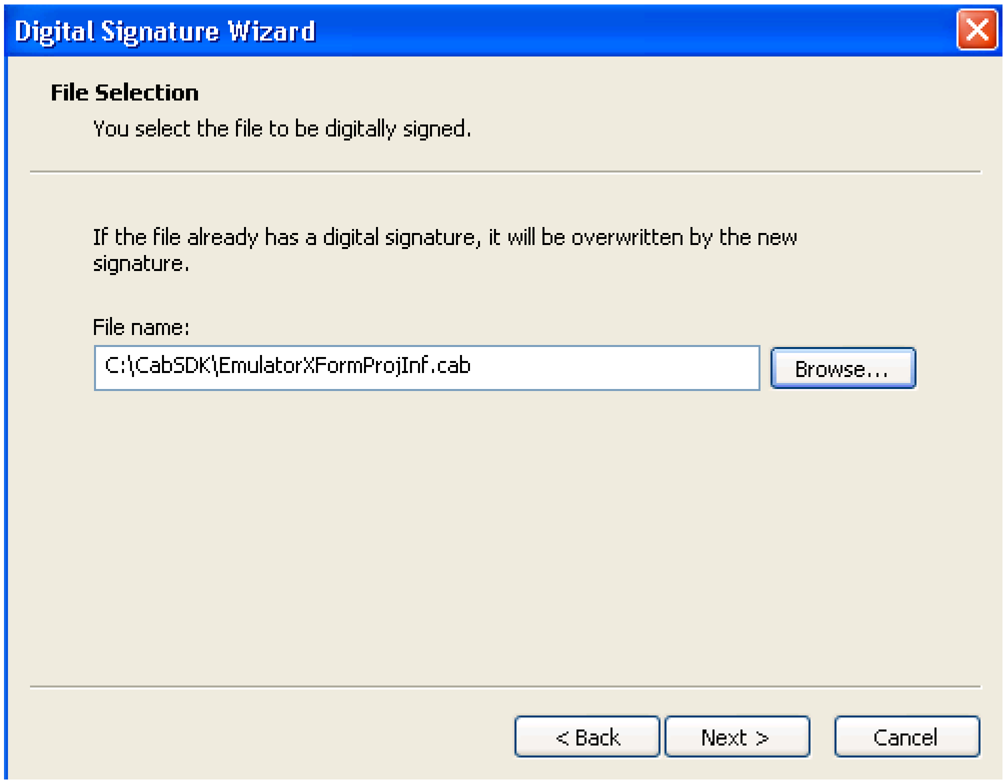
Welcome to the Digital Signature Wizard

This wizard helps you attach a digital signature to a file.

A digital signature verifies that no changes have been made to the file.

To continue, click Next.

Press the Next > button to continue the Digital Signature process.

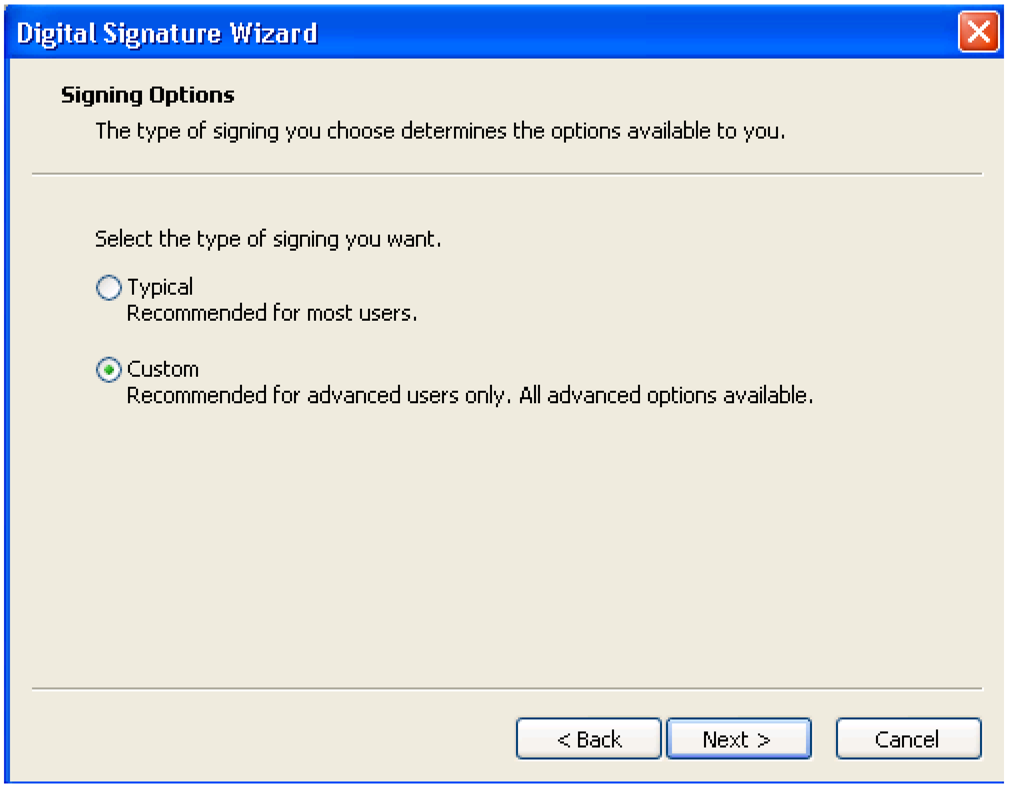


File Selection
You select the file to be digitally signed.

If the file already has a digital signature, it will be overwritten by the new signature.

File name:
C:\CabSDK\EmulatorXFormProjInf.cab

In this dialog, locate your ActiveX control (.ocx) or in the example's case, your Microsoft Cabinet file. This is the application you are signing. Many times you will have to sign multiple cabinet files. Press Next > to continue:



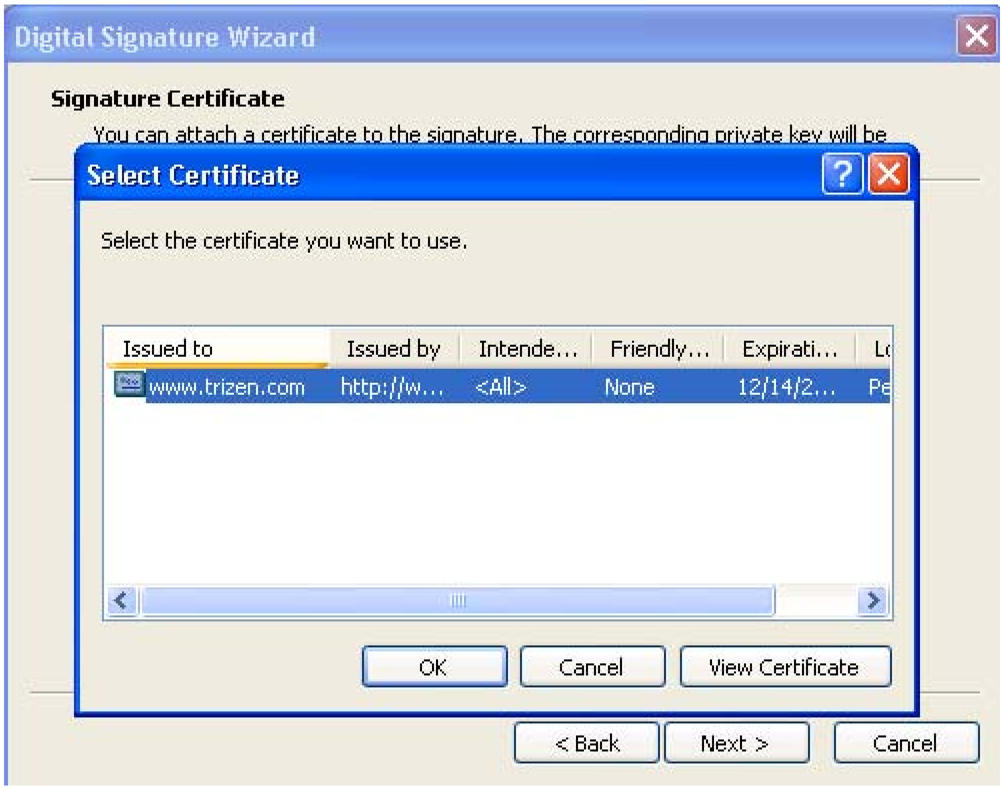
Signing Options
The type of signing you choose determines the options available to you.

Select the type of signing you want.

Typical
Recommended for most users.

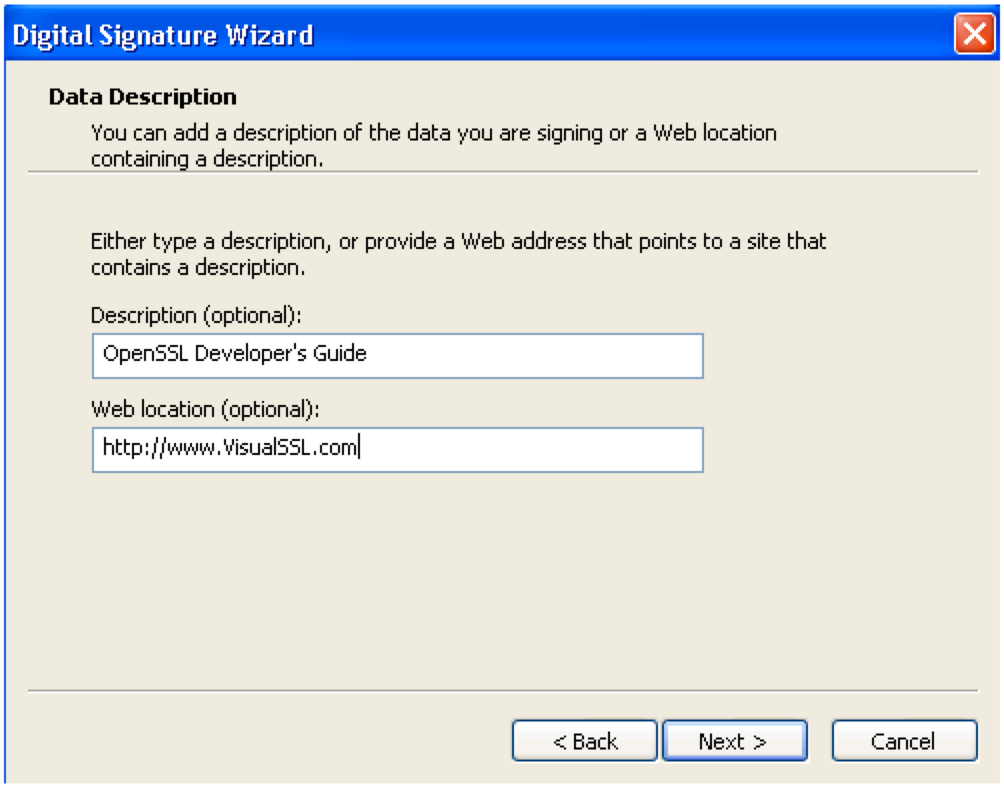
[✓ Custom]
Recommended for advanced users only. All advanced options available.

Since our certificate is already contained in our Personal Certificate store, choosing either the Typical or Custom option will have the same result; however, for those that are importing a file, you must use the Custom option. After pressing the Next > button you should press the "Select from Store" button (this will be the only option if you selected Typical).


Digital Signature Wizard
Signature Certificate
You can attach a certificate to the signature. The corresponding private key will be
Select Certificate
Select the certificate you want to use.
OK
Cancel
View Certificate
< Back
Next >
Cancel

Issued to	Issued by	Intende...	Friendly...	Expirati...	Lo
www.trizen.com	http://w...	<All>	None	12/14/2...	Pe

If all has gone well you should see the PKCS#12 certificate imported earlier. Select this certificate by pressing the OK button. If you did not see this in your dialog, make sure you have followed the directions exactly as outlined. Press Next > to continue the code signing process.



Data Description

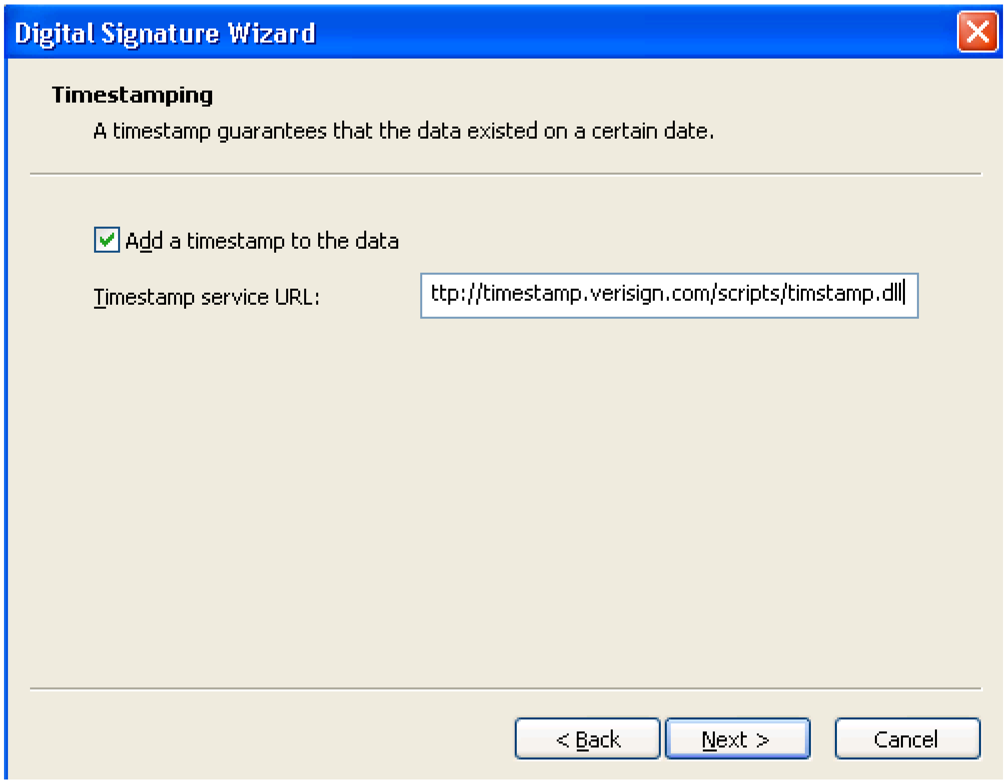
You can add a description of the data you are signing or a Web location containing a description.

Either type a description, or provide a Web address that points to a site that contains a description.

Description (optional): OpenSSL Developer's Guide

Web location (optional): http://www.VisualSSL.com

When your Authenticode certificate is presented to the end user, the information that you type into the above dialog will also be presented with a hypertext link to your site. Press Next > to continue.



Timestamping

A timestamp guarantees that the data existed on a certain date.

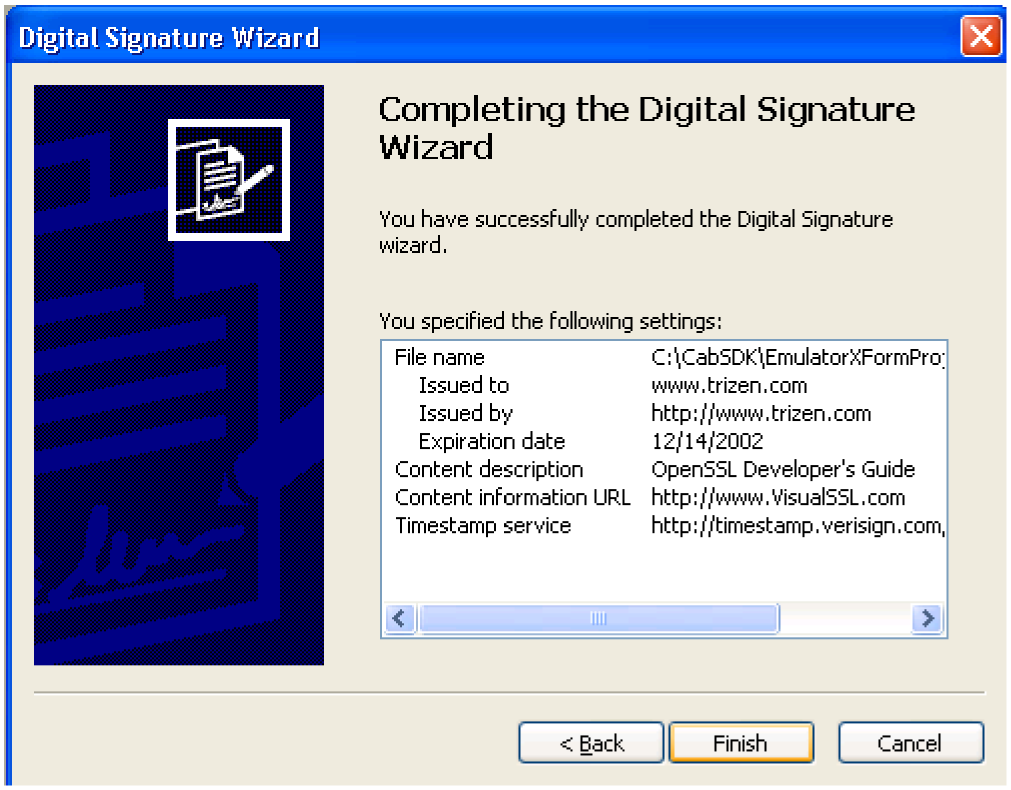
[✓] Add a timestamp to the data

Timestamp service URL: ttp://timestamp.verisign.com/scripts/timstamp.dll

The timestamp dialog offers you an opportunity to timestamp your certificates to differentiate them if necessary and discourage forgeries. This is completely optional and if desired type the following into the edit box:

http://timestamp.verisign.com/scripts/timstamp.dll

After pressing the Next > button you will be presented with the following dialog which will complete the code signing process:



Completing the Digital Signature Wizard

You have successfully completed the Digital Signature wizard.

You specified the following settings:

File name	C:\CabSDK\EmulatorXFormPro
Issued to	www.trizen.com
Issued by	http://www.trizen.com
Expiration date	12/14/2002
Content description	OpenSSL Developer's Guide
Content information URL	http://www.VisualSSL.com
Timestamp service	http://timestamp.verisign.com,

Press the Finish button to complete the process. Your code has now been Digitally signed by your Certificate Authority. Remember this is not the only step to make your controls simple for download use when Internet Explorer's security settings are high. For an in depth look at creating ActiveX controls for use over the Internet check out our website at http://www.trizen.com.

Summary

OpenSSL is a great implementation and as far as an open source product goes, offers a great toolkit! This document is intended for use with the Visual SSL product from Trizen Systems which offers custom SSL components utilizing the OpenSSL libraries for Delphi and Borland C++ Builder. It was our intent **not** to show the theory behind SSL, because as anyone can tell you there is a lot of theory on the subject, but rather to demonstrate how SSL is actually used within Windows. If you would like to get more information please visit http://www.openssl.org for a good bit of information as well as looking in the C:\OpenSSL\openssl-0.9.6a\docs directory for information on commands and applications.

It goes without saying, but by only experimenting with SSL can you really learn about what the product can accomplish. We highly suggest you play with the switches for the *s_client* and *s_server* applications.

We hope you will carry the torch for the OpenSSL project and implement SSL into all of your applications. For implementation examples you can purchase the Visual SSL source code and/or the Visual 3270 source code for your review.

Index