Customizing X.509 Certificate Fields

Charles D. Short

CS526 – S2008 - Semester Project

University of Colorado at Colorado Springs
Dr. C Edward Chow

Abstract

An X.509 certificate can be used for authentication between a client and server to insure client identity but does not generally provide any additional fields for information which may be useful to custom applications running on the server. This paper will detail how to insert additional information into a client certificate which may then be used by a server based application to provide services or make decisions based upon this information. The examples presented in this paper are built using VMware Server software to host an Ubuntu Linux server running Apache and PHP. The browser used for the certificate demonstration is Microsoft Internet Explorer.

Introduction

X.509 certificates are in wide use throughout the internet community today and used primarily for the identification and verification of host server identity to client users. However, it is surprising how little information is available related to their capabilities for use beyond basic authentication. Especially in regards to their extensibility for providing additional information to host based server applications to customize services based upon information contained in the certificate itself.

This paper details a method for including additional information into a certificate Subject Distinguished Name field, signing the certificate and installing it on a client browser. I then demonstrate how to configure Apache to request the client to present the certificate to the host server. The Subject Distinguished Name information is then extracted via a simple PHP script and displayed in the client browser for demonstration purposes.

In addition, the Apache configuration for providing a secure SSL connection to the client browser is included.

Discussion

X.509 Background

The International Telecommunications Union published the beginnings of the X.509 standard in conjunction with the X.500 standard circa 1988. Since that time the standard has been enhanced to provide functionality necessary for the internet environment by the Internet Engineering Task Force (IETF) Public-Key Infrastructure (X.509) working group, or commonly referred to as the PKIX working group. Currently, the term *X.509 certificate* generally refers to the PKIX Certificate and Certificate Revocation List Profile of the X.509 standard as specified in IETF RFC3280.

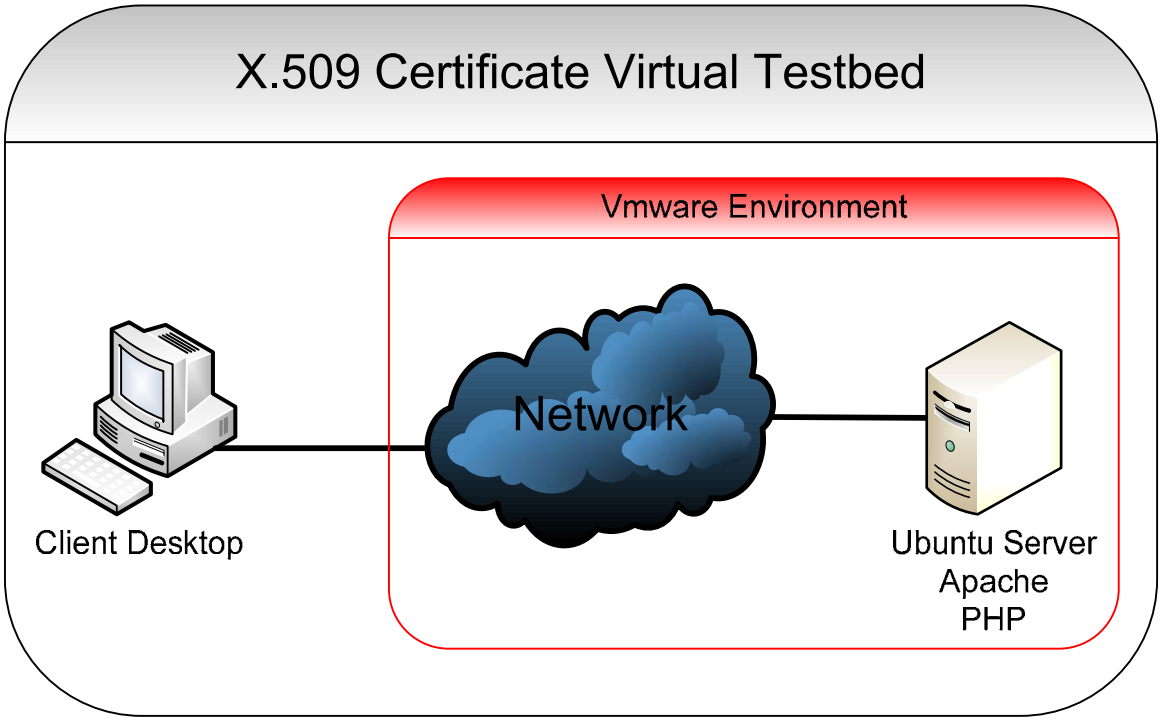
OpenSSL Background

OpenSSL is an Open Source project (http://www.openssl.org/) based upon the SSLeay library developed by Eric A. Young and Tim J. Hudson. It provides a comprehensive cryptographic tool kit for implementing both Secure Sockets Layer (SSL) and Transport Security Layer (TLS) protocols, as well as, the creation and management of X.509 digital certificates. Many examples of its usage are presented throughout this paper.

X.509 Virtual Test Environment

The X.509 test environment used for this project consisted of VMware server software running on a laptop computer. The VMware server hosts an Ubuntu server instance where OpenSSL, Apache and PHP were installed to simulate a live server environment. The client laptop browser was then used to communicate with the virtual server environment. This configuration is detailed in Figure (1) below.

Figure (1):



OpenSSL Commands

There are several OpenSSL commands provided specifically for the creation and manipulation of X.509 certificates. A brief synopsis of these is included below. For further information, each is described in great detail within the Ubuntu man pages by typing *man <command name>*.

ca – OpenSSL Certificate Authority command. It is used for signing certificate requests, generating certificate revocation lists and maintaining the certificate authorities certificate database.

req – OpenSSL command for creating and processing certificate requests. It is used to create certificate requests and display certificate fields. It can also be used to create self signed certificates for testing purposes.

x509 –Mulit-purpose openSSL command. It can be used for displaying certificate information, converting certificates to various forms, signing certificate requests or editing certificate trust settings.

OpenSSL Certificate Configuration Files

Associated with the various OpenSSL commands for creating and signing certificates is a configuration file generally named ***<filename>.cnf***. This file is used to provide information for use by the OpenSSL command. Much of this information can also be provided explicitly using OpenSSL command line options.

A default version of this file (*opennssl.cnf)* is provided with the Ubuntu distribution in */etc/ssl/oopenssl.cnf* and is included in Appendix (5) of this document. Please note, I have modified the default version of this file for the creation of the Certificate Authority, Server, and Client certificates necessary for this project. These configuration files are also included in this document as Appendices (1), (2), and (3) respectively.

OpenSSL configuration files consists of sections which begin with bracketed identifiers, such as **[ca**] to identify the beginning of a section. Each section then corresponds to an OpenSSL command **ca**, **req**, or **x509**. Within sections a set of directives of the form **<attribute> = <value>,** provide information which influence the operation of the particular OpenSSL command.

A good definition of the *openssl.cnf* file sections and attributes can be found at this web site:

http://www.technoids.org/openssl.cnf.html

Object Identifier (OID)

According to the default openssl.cnf file it is necessary to establish an OID number for each new field and value included in a certificate. Based upon testing I found this to be true. Therefore, prior to adding additional information to a certificate it is important to understand the concept of the Object Identifier (OID). I discovered the following IETF RFC's which outline the OID numbering system.

IETF RFC 1778 details OID syntax in section 2.15 http://www.ietf.org/rfc/rfc1778.txt.

IETF RFC3061 defines a set of OIDs as a tree where each leaf is simply a sequence of digits. http://www.ietf.org/rfc/rfc3061.txt In this document *OID 1.3.6.1* is stated to be the *Internet OID*.

Unfortunately, I was not able to locate a definitive central global authority for registering OID information during the course of this project. In fact, all of the information I reviewed implied there was no central body.

However, I did discover several registration authorities of which I will mention two.

1) The Internet Assigned Number Authority (IANA). Under the Protocol Registries section http://www.iana.org/assignments an entry labeled Enterprise Numbers which provides for registration of OIDs as Private Enterprise Numbers under the prefix of iso.org.dod.internet.private.enterprise (1.3.6.1.4.1). Since the OID, by definition, can be extended once it is assigned, this appears to be the most likely location for registration of a new OID for a specific organization. These OID/Enterprise numbers are registered using the link: http://pen.iana.org/pen/PenApplication.page

2) The ITU ANS.1 site http://www.itu.int/ITU-T/asn1/ provides a link to generate and register http://www.itu.int/ITU-T/asn1/uuid.html a Universally Unique Identifier (UUID) as an OID in their database.

   This UUID was generated using the site: 0de9ac80-1d2b-11dd-9236-0002a5d5c51b

In addition, RFC4043 provides some information regarding the registration of OID information http://www.ietf.org/rfc/rfc4043.txt.

Please note: for the purpose of this project, I chose to use ***OID numbers 1.2.3.4 and 1.2.3.5***. Since, my certificates will only exist within my virtual experimentation environment, it is not necessary to insure uniqueness through an acquisition/registration process.

Abstract Syntax Notation - ASN.1

The next concept necessary for understanding certificates is to be aware of Abstract Syntax Notation One (ASN.1).

As described on the ASN.1 web page http://asn1.elibel.tm.fr/en/introduction/index.htm "*ASN.1 is a formal notation used for describing data transmitted by telecommunications protocols, regardless of language implementation and physical representation of these data, whatever the application, whether complex or very simple*."

The importance of this is that ASN.1 is the language/syntax that describes the structure and contents of a certificate in its digital form. In order to understand this relationship see IETF RFC 3280 at http://www.ietf.org/rfc/rfc3280.txt. This RFC defines the fields and attributes of a certificate using ANS.1.

Adding New Client Subject Field Information

Certificate Authority openssl.cnf Modifications

Following are the openssl.cnf additions which were necessary to prepare the Certificate Authority for signing the client request with the new field information.

1) Insert a **<name = OID value>** pair for each new field in the **[new_oids]** section

   I chose *NewField1* and *NewField2* to be the new field names with *1.2.3.4* and *1.2.3.5* as their respective OIDs. Adding the information in this section enables openssl to refer to the new fields by name throughout the remainder of the openssl.cnf file.

2) Add the **<name = policy>** pair of each field in the **[policy_match]** section

   I chose to use the *optional* policy value so this opoenssl.cnf file can also be used to sign certificates which do not contain these added fields.

A snippet of the CA openssl.cnf file is included as below. The entirety of the CA openssl.cnf file is included in Appendix (1).

Certificate Authority (CA) OpenSSL Configuration File snippet

```
.
.
.
oid_section = new_oids

[ new_oids ]
```

```
NewField1 = 1.2.3.4
NewField2 = 1.2.3.5
policy = policy_match
.
.
.
[ policy_match ]
.
.
.
NewField1 = optional
NewField2 = optional
.
.
.
```

Client openssl.cnf Modifications

Following are the openssl.cnf additions which were necessary for the client certificate request.

1) Insert a **<name = OID value>** pair for each new field in the **[new_oids]** section

   I chose *NewField1* and *NewField2* to be the new field names with *1.2.3.4* and *1.2.3.5* as their respective OIDs. Adding the information in this section enables openssl to refer to the new fields by name throughout the remainder of the openssl.cnf file.

2) Add the **<name = value>** pair of each field in the **[policy_match]** section

   The first <name = value pair>, **NewField1 = New certificate field 1**, sets the text to be displayed at the prompt when creating the certificate request with OpenSSL. The second, <name = value pair>, **NewField1_default = New certificate field 1**, sets the value of NewField1 to be "New certificate field 1" by default.

A snippet of the Client openssl.cnf file is included as below. The entirety of the Client openssl.cnf file is included in Appendix (3).

Client OpenSSL Configuration File snippet

```
.
.
.
oid_section = new_oids

[ new_oids ]

NewField1 = 1.2.3.4
NewField2 = 1.2.3.5
```

```
.
.
.
[ req ]

distinguished_name = client_distinguished_name
.
.
.
[ client_distinguished_name ]

#Add new fields to the Certificate.

#Text to display at prompt
NewField1 = New certificate field 1

#Default field contents
NewField1_default = New certificate field 1

#Text to display at prompt
NewField2 = New certificate field 2

#Default field contents
NewField2_default = New certificate field 2
.
.
.
```

Create a Certificate Authority

1) Create working directories to be used by the Certificate Authority. The directory myCA will contain the CA certificate, certificate database, generated certificates, keys, and certificate requests. The directory myCA/signedcerts will contain copies of each signed certificate.

```
mkdir myCA
mkdir myCA/signedcerts
cd myCA
```

2) Create certificate database files. The serial file will contain the index number of the next certificate beginning at 1. The index.txt file will contain each signed certificate indexed by the index number value.

```
echo '01' > serial
touch index.txt
```

3) Generate a self-signed CA public certificate and private key:

```
openssl req -config certreq.cnf -x509 -newkey rsa:2048 –keyout cakey.pem -out cacert.pem
```

Create a Server certificate

1) Generate the server certificate request and private key

```
openssl req -config serverreq.cnf -newkey rsa:1024 –keyout servertempkey.pem -out serverreq.pem
```

2) Remove the password from the server key

```
openssl rsa < servertempkey.pem > serverkey.pem
```

3) Sign the server certificate using the Certificate Authority

```
openssl ca -config certreq.cnf -in serverreq.pem -out servercrt.pem
```

Create the client certificate

1) Generate the client certificate request and private key

```
openssl req -config clientreq.cnf -newkey rsa:1024 -out clientreq.pem -keyout clientkey.pem
```

2) Display the Client certificate request fields

```
openssl asn1parse -in clientreq.pem
```

3) Sign the client certificate using the Certificate Authority

```
openssl ca -config certreq.cnf -in clientreq.pem -out clientcrt.pem
```

4) Combine the clientkey.pem and clientcrt.pem

```
cat clientkey.pem clientcrt.pem > clientcrtandkey.pem
```

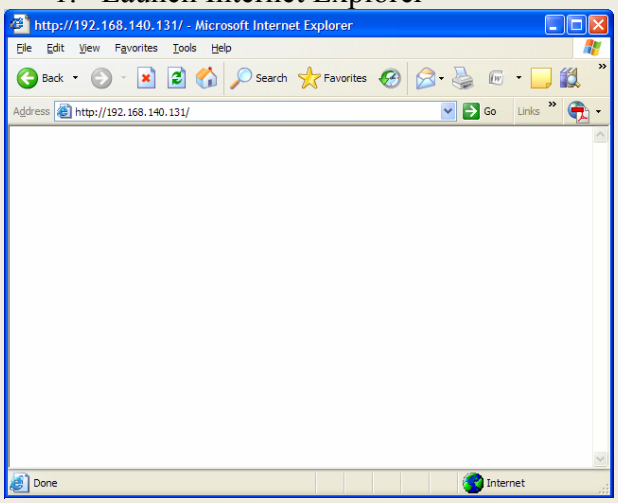
5) Convert X.509 Certificate to PKCS#12

```
openssl pkcs12 -export -out clientcert.pfx -in clientcrtandkey.pem -name "UbuntuWS1 Client certificate”
```

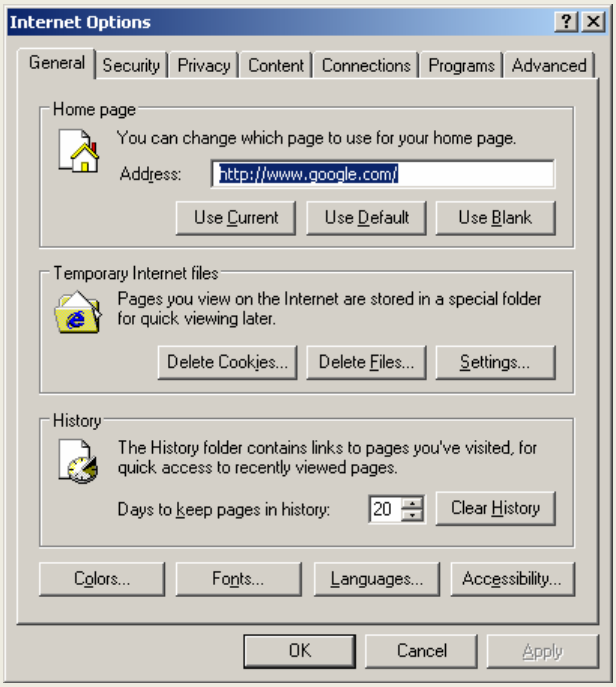
Install Client Certificate on Internet Explorer

The following screen images detail installing the client certificate using Microsoft Internet Explorer.

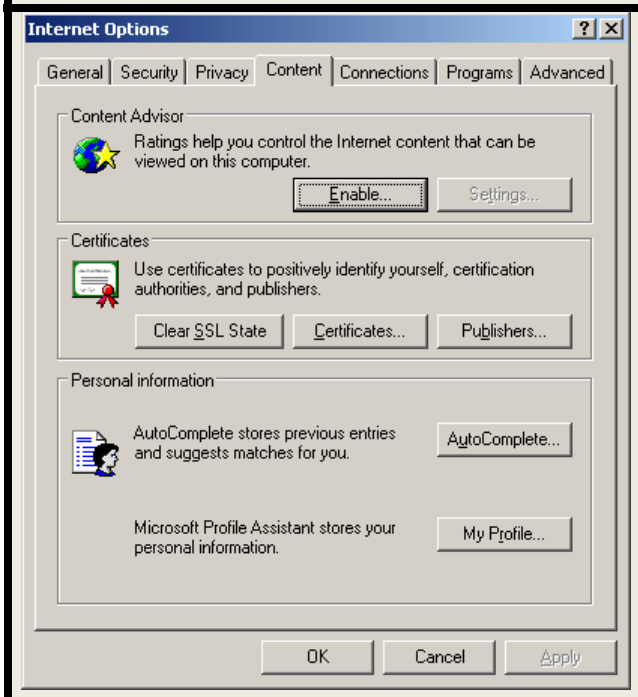
1. Launch Internet Explorer



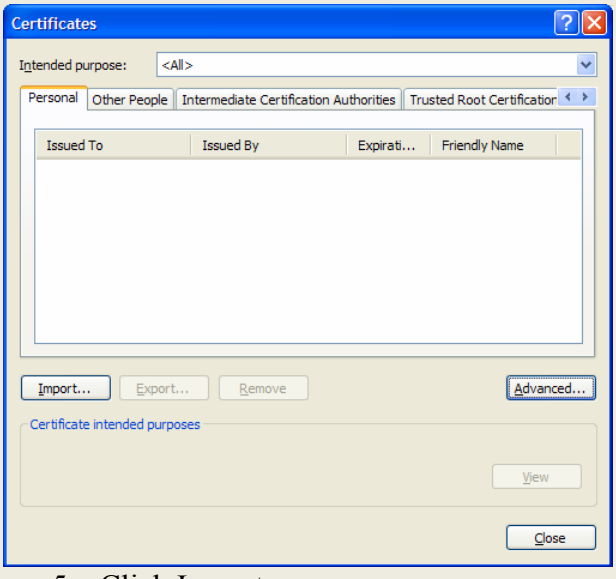
2. Click Tools/Internet Options…



3. Click Content

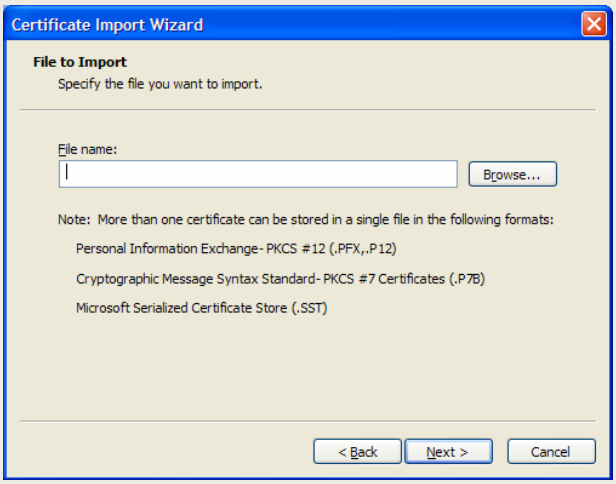


4. Click Certificates…

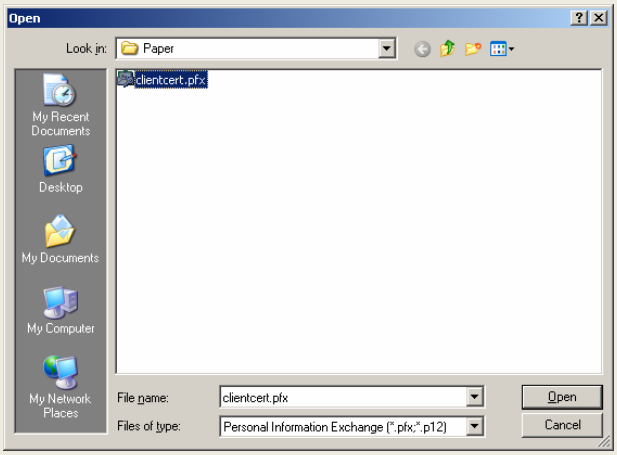


5. Click Import…

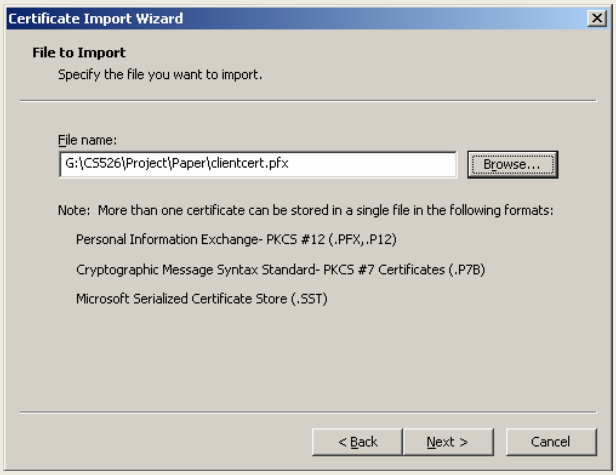
6. Click Next



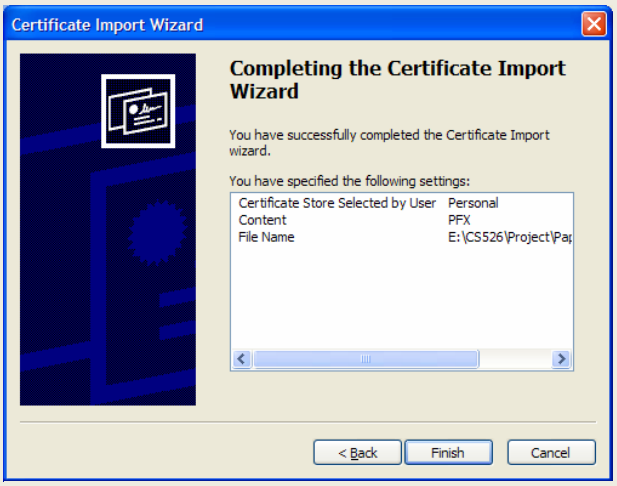
7. Click Browse…



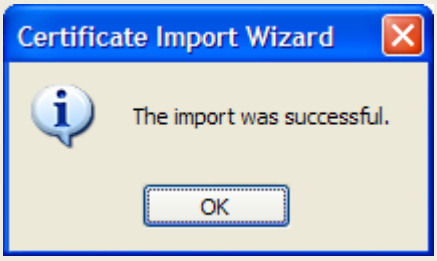
8. Click Open



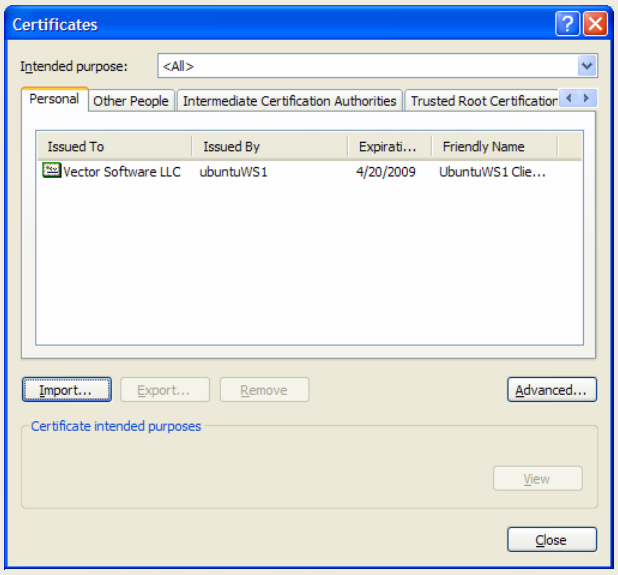
9. Click Next



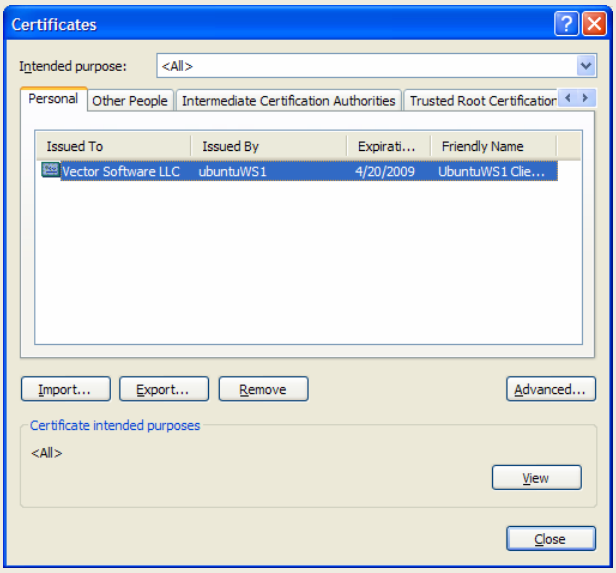
10. Click Finish



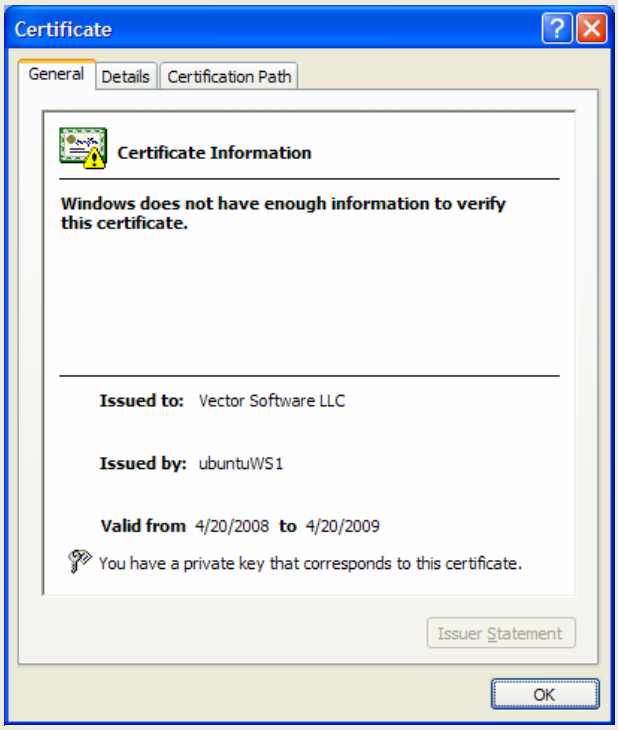
11. Click Ok



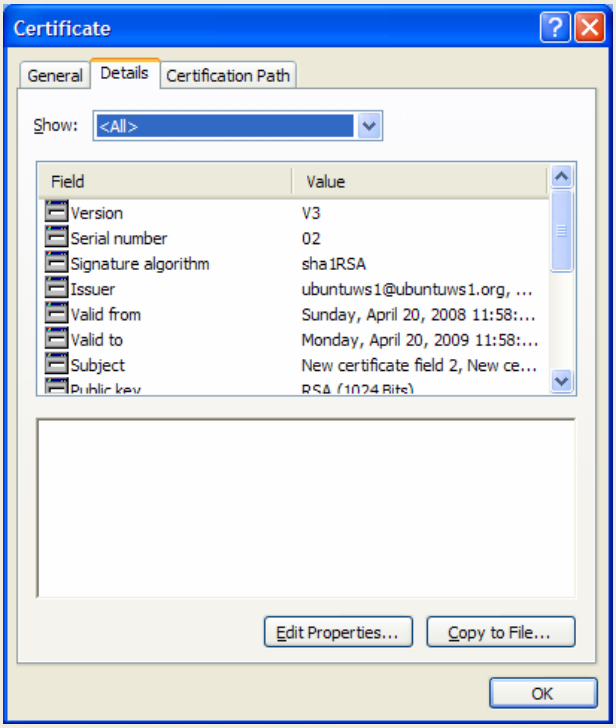
12. Select Certificate



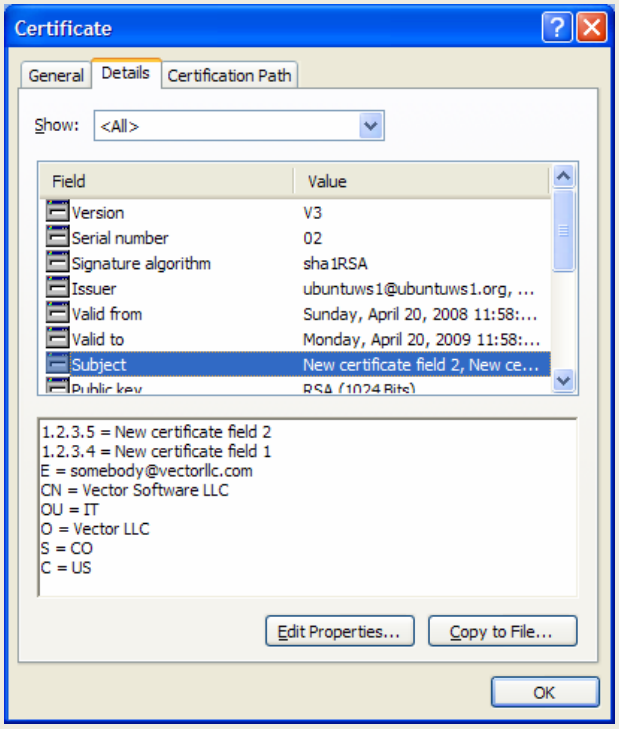
13. Click View



14. Click Details



15. Select Subject field



← Notice the new OID's 1.2.3.4 and 1.2.3.5 with their associated field information have been included as part of the subject in the client certificate. Also notice that the field name information: NewField1 and NewField2 are not included.

Setup Ubuntu Apache Web Server to Test certificate

1) Enable SSL on the Web Server

```
sudo a2enmod ssl
```

2) Move the server certificate and key to the appropriate locations

```
sudo cp servercrt.pem /etc/ssl/certs
sudo cp serverkey.pem /etc/ssl/private
```

3) Move the CA certificate to the appropriate location

```
sudo cp cacert.pem  /etc/ssl/certs
```

4) Add the following lines to the */etc/apache2/sites-available/default* file in the virtual host section under the DocumentRoot line to inform the server to enable SSL and tell it where the certificates and keys reside

```
SSLEngine on
SSLCertificateFile /etc/ssl/certs/servercrt.pem
SSLCertificateKeyFile /etc/ssl/private/serverkey.pem
```

5) Add the following line to the */etc/apache2/ports.conf* file to inform the server which port to enable SSL on

```
Listen 443
```

6) Insert these lines into /etc/apache2/httpd.conf to tell the server to request a certificate from the Client, the CA certificate used to sign the Client certificate should be found locally and, finally, where to find it

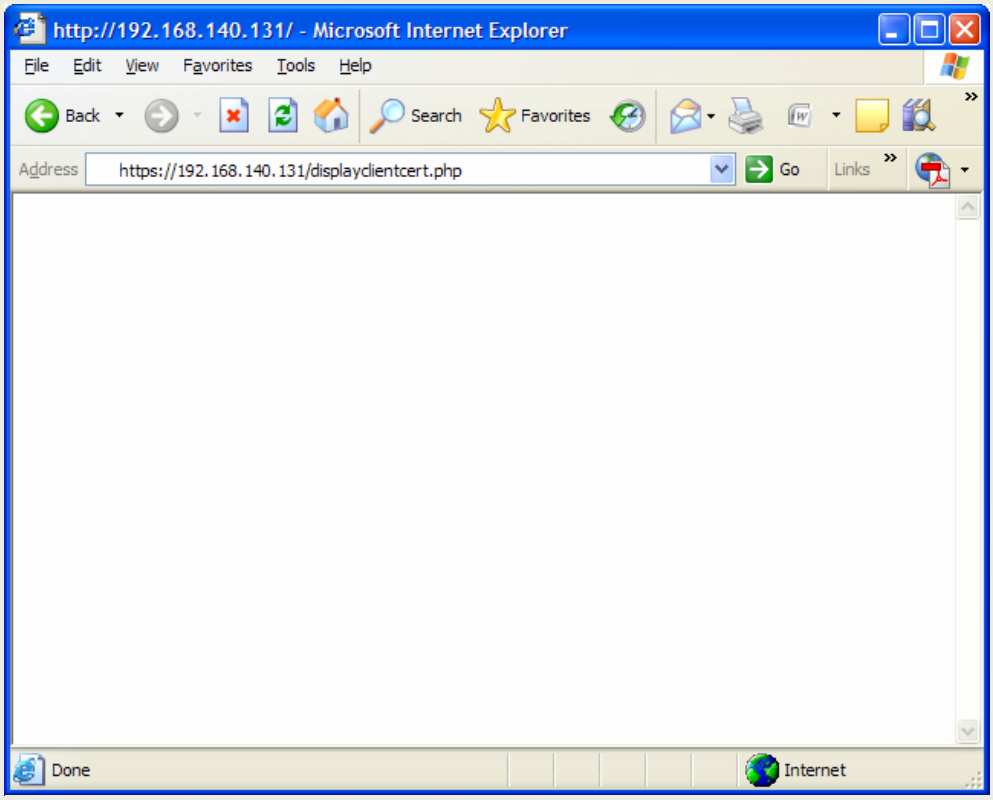
```
SSLVerifyClient require
SSLVerifyDepth 1
SSLCACertificateFile /etc/ssl/certs
```

7) Restart the web server to make the changes take effect

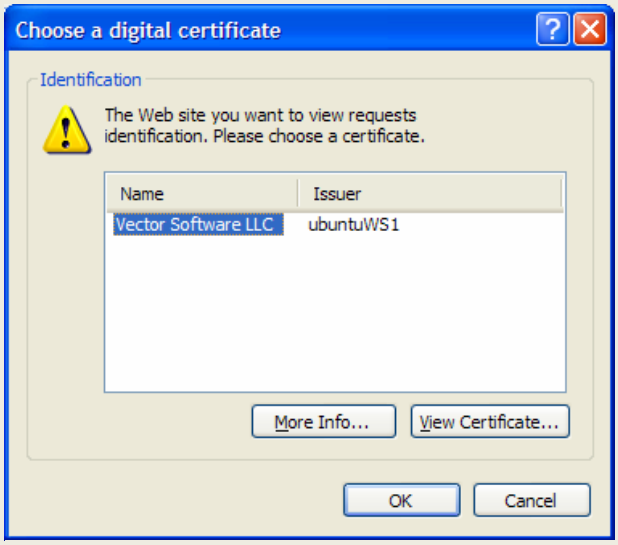
```
sudo /etc/init.d/apache2 restart
```

Test the Client and Server Certificates

The following series of screen images illustrates the client and server interaction with each other using the certificates I created.



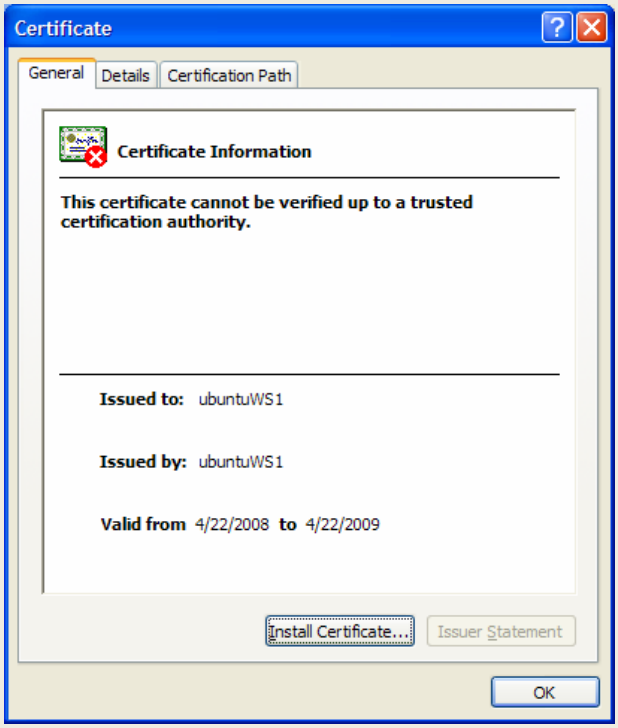
1. Connect to the server using https



Notice that the server requests the client certificate for authentication.

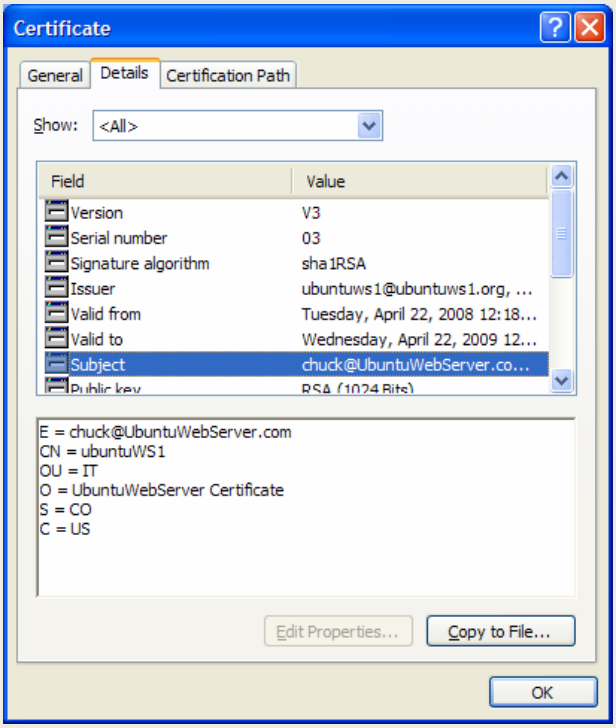
2. Click Ok

3. Click View Certificate



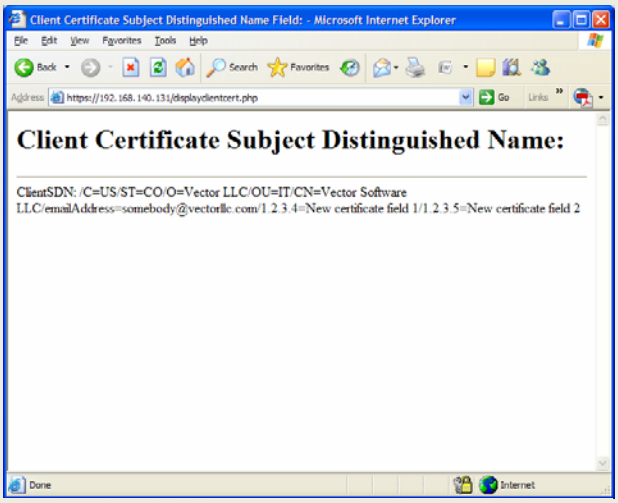
Notice the server has accepted the client certificate and has responded with its certificate to start the HTTPS session.

4. Click Details



5. Click Ok

6. Click Yes



7. Notice the server correctly reports the client Subject Distinguished Name information including the two fields that have been added. Please note the fields are reported by OID number rather than the names I had given (i.e. NewField1 and NewField2). The server side PHP script used for this demonstration is included in Appendix(4) of this document.

Lessons Learned

The documentation associated with the creation, management, and modification of X.509 certificates is scarce and the documentation which is readily available is confusing. Even the names of the fields contained in a certificate are sometimes difficult to relate to real world applications. In order to grasp the meaning of much of the certificate process, it is necessary to refer to many sources of information which proved to be a cumbersome and confusing process.

Moreover, the flexibility provided with the current certificate format is limited. Information included in a certificate, such as that detailed in this document, is not readily usable in a browser without modification of the browser to understand the meaning and names of new fields and information. As illustrated in this project, neither the browser nor the Apache MOD_SSL is aware that 1.2.3.4 is NewField1 and 1.2.3.5 is NewField2. It would be more convenient if certificate field name information were communicated as part of the certificate itself rather than as a predefined object.

Future Work

In the future I would like to explore the possibility of adding additional fields outside of the Subject Distinguished Name section of the certificate. This would include exploring the creation of a different certificate format which includes field name information as part of the certificate itself. This would provide a convenient means to communicate information securely between systems since the information source certificate and all of its contents could still be signed by a trusted CA insuring their validity. I would also like to explore the modification of MOD_SSL and a client browser to recognize the new certificate format.

References

http://cs.uccs.edu/~cs526/secureWebAccess/secureWebAccess.htm
http://httpd.apache.org/docs/2.2/mod/mod_ssl.html
http://www.itu.int/ITU-T/asn1/
http://www.modssl.org/docs/2.8/ssl_howto.html#ToC6
http://www.oid-info.com/index.htm
http://www.oid-info.com/standards.htm
http://www.openssl.org/
http://www.openssl.org/docs/apps/ca.html
http://www.openssl.org/docs/apps/req.html
http://www.openssl.org/docs/apps/req.html#CONFIGURATION_FILE_FORMAT
http://www.openssl.org/docs/apps/x509.html
http://www.technoids.org/openssl.cnf.html
http://www.zaphu.com/2007/08/21/ubuntu-lamp-server-guide-configure-apache-mysql-and-cgi-bin/
https://help.ubuntu.com/6.06/ubuntu/serverguide/C/httpd.html
https://help.ubuntu.com/community/OpenSSL
http://en.wikipedia.org/wiki/X.509
http://www.ietf.org/rfc/rfc3061.txt
http://www.itu.int/ITU-T/asn1/
http://www.ietf.org/rfc/rfc4043.txt
http://www.ietf.org/rfc/rfc3280.txt
http://asn1.elibel.tm.fr/en/introduction/index.htm

Abstract Syntax Notation One (ASN.1) **Specification of Basic Notation** ITU-T Rec. X.680 (2002) | ISO/IEC 8824-1:2002

Ubuntu openssl man pages

Appendix (1): CA OpenSSL configuration file

```
#
# CA OpenSSL configuration file
#

oid_section                = new_oids

[ new_oids ]

#Assign Object Identiifiers for new field names.

NewField1 = 1.2.3.4
NewField2 = 1.2.3.5

[ ca ]
default_ca      = CA_default            # The default ca section

[ CA_default ]

dir             = /home/chuck/myCA      # Where everything is kept
certs           = $dir                  # Where the issued certs are kept
crl_dir         = $dir          # Where the issued crl are kept
database        = $dir/index.txt        # database index file.
#unique_subject         = no                    # Set to 'no' to allow creation of
                                        # several ctificates with same subject.
new_certs_dir   = $dir/signedcerts      # default place for new certs.

certificate     = $dir/cacert.pem       # The CA certificate
serial          = $dir/serial   # The current serial number
crlnumber       = $dir                  # the current crl number
                                                # must be commented out to leave a V1 CRL
crl             = $dir                  # The current CRL
private_key     = $dir/cakey.pem                # The private key
RANDFILE        = $dir                  # private random number file

x509_extensions         = usr_cert              # The extentions to add to the cert

# Comment out the following two lines for the "traditional"
# (and highly broken) format.
name_opt        = ca_default            # Subject Name options
cert_opt        = ca_default            # Certificate field options

# Extension copying option: use with caution.
# copy_extensions = copy

# Extensions to add to a CRL. Note: Netscape communicator chokes on V2 CRLs
# so this is commented out by default to leave a V1 CRL.
# crlnumber must also be commented out to leave a V1 CRL.
# crl_extensions= crl_ext
```

```
default_days     = 365                  # how long to certify for
default_crl_days= 365                   # how long before next CRL
default_md       = sha1                 # which md to use.
preserve         = no                           # keep passed DN ordering

policy           = policy_match

[ policy_match ]
countryName              = match
stateOrProvinceName      = match
organizationName         = optional
organizationalUnitName           = optional
commonName               = supplied
emailAddress             = optional
NewField1            = optional
NewField2            = optional

# For the 'anything' policy
# At this point in time, you must list all acceptable 'object'
# types.

[ policy_anything ]
countryName                      = optional
stateOrProvinceName      = optional
localityName             = optional
organizationName                 = optional
organizationalUnitName           = optional
commonName                       = supplied
emailAddress             = optional

[ req ]
default_bits             = 1024
default_keyfile          = privkey.pem
distinguished_name       = req_distinguished_name
attributes                       = req_attributes
x509_extensions                  = v3_ca

# Passwords for private keys if not present they will be prompted for
input_password = chuck
output_password = chuck

string_mask = nombstr

[ req_distinguished_name ]
countryName                              = Country Name (2 letter code)
countryName_default              = US
countryName_min                          = 2
countryName_max                          = 2

stateOrProvinceName              = State or Province Name (full name)
```

```
stateOrProvinceName_default    = CO

localityName                   = Locality Name (eg, city)
localityName_default           = CS

organizationName                       = Organization Name (eg, company)
organizationName_default       = Certificate Authority

organizationalUnitName                         = Organizational Unit Name (eg, section)
organizationalUnitName_default         = IT

commonName                             = Common Name (eg, YOUR name)
commonName_max                         = 64
commonName_default             = ubuntuWS1

emailAddress                   = Email Address
emailAddress_max                       = 64
emailAddress_default           = ubuntuws1@ubuntuws1.org

[ req_attributes ]
challengePassword              = A challenge password
challengePassword_min          = 4
challengePassword_max          = 20

[ usr_cert ]

basicConstraints=CA:FALSE

nsComment= "OpenSSL Generated Certificate"

subjectKeyIdentifier=hash
authorityKeyIdentifier=keyid,issuer

[ v3_req ]

basicConstraints = CA:FALSE
keyUsage = nonRepudiation, digitalSignature, keyEncipherment

[ v3_ca ]

subjectKeyIdentifier=hash

authorityKeyIdentifier=keyid:always,issuer:always

basicConstraints = CA:true

[ crl_ext ]

authorityKeyIdentifier=keyid:always,issuer:always
```

Appendix (2): Server OpenSSL configuration file

```
#
# Server certificate request configuration file
#

[ req ]
prompt                = no
distinguished_name    = server_distinguished_name

[ server_distinguished_name ]
commonName            = UbuntuWS1
stateOrProvinceName   = CO
countryName           = US
emailAddress          = chuck@UbuntuWebServer.com
organizationName      = UbuntuWebServer Certificate
organizationalUnitName  = IT
```

Appendix (3): Client OpenSSL configuration file

```
#
# Client OpenSSL configuration file.
#

oid_section = new_oids

[ new_oids ]

#Assign Object Identiifiers for new field names.

NewField1 = 1.2.3.4
NewField2 = 1.2.3.5

[ req ]

distinguished_name = client_distinguished_name

# Passwords for private keys if not present they will be
prompted for
input_password  = chuck
output_password = chuck

string_mask = nombstr

[ client_distinguished_name ]
countryName                     = Country Name (2 letter code)
countryName_default             = US
countryName_min                 = 2
countryName_max                 = 2

stateOrProvinceName             = State or Province Name (full
name)
stateOrProvinceName_default     = CO

localityName                    = Locality Name (eg, city)
localityName_default            = CS
organizationName                = Organization Name (eg,
company)
organizationName_default        = Vector LLC

organizationalUnitName          = Organizational Unit Name (eg,
section)
organizationalUnitName_default  = IT

commonName                      = Common Name (eg, YOUR name)
```

```
commonName_max                  = 64
commonName_default              = Vector Software LLC

emailAddress                    = Email Address
emailAddress_max                = 64
emailAddress_default            = somebody@vectorllc.com

#Add new fields to the Certificate.

#Text to display at prompt
NewField1 = New certificate field 1

#Default field contents
NewField1_default = New certificate field 1

#Text to display at prompt
NewField2 = New certificate field 2

#Default field contents
NewField2_default = New certificate field 2
```

Appendix (4): Server PHP Script (/var/www/displayclientcert.php)

```
<!-- Display Client Certificate Fields -->
<head>
<title>Client Certificate Subject Distinguished Name Field:</title>
</head>
<body>
<h1><b>Client Certificate Subject Distinguished Name:</b></h1>
<hr>
<?php
   print("ClientSDN: $_SERVER[SSL_CLIENT_S_DN]<br><br>");
?>
```

Appendix (5): Default OpenSSL 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 ]

dir             = ./demoCA              # Where everything is
kept
certs           = $dir/certs            # Where the issued certs
are kept
```

```
crl_dir         = $dir/crl              # Where the issued crl
are kept
database        = $dir/index.txt        # database index file.
#unique_subject = no                    # Set to 'no' to allow
creation of
                                        # several ctificates
with same subject.
new_certs_dir   = $dir/newcerts         # default place for new
certs.

certificate     = $dir/cacert.pem       # The CA certificate
serial          = $dir/serial           # The current serial
number
crlnumber       = $dir/crlnumber        # the current crl number
                                        # must be commented out
to leave a V1 CRL
crl             = $dir/crl.pem          # The current CRL
private_key     = $dir/private/cakey.pem# The private key
RANDFILE        = $dir/private/.rand    # private random number
file

x509_extensions = usr_cert              # The extentions to add
to the cert

# Comment out the following two lines for the "traditional"
# (and highly broken) format.
name_opt        = ca_default            # Subject Name options
cert_opt        = ca_default            # Certificate field
options

# Extension copying option: use with caution.
# copy_extensions = copy

# Extensions to add to a CRL. Note: Netscape communicator chokes
on V2 CRLs
# so this is commented out by default to leave a V1 CRL.
# crlnumber must also be commented out to leave a V1 CRL.
# crl_extensions        = crl_ext

default_days    = 365                   # how long to certify
for
default_crl_days= 30                    # how long before next
CRL
default_md      = sha1                  # which md to use.
preserve        = no                    # keep passed DN
ordering
```

```
# A few difference way of specifying how similar the request
should look
# For type CA, the listed attributes must be the same, and the
optional
# and supplied fields are just that :-)
policy          = policy_match

# For the CA policy
[ policy_match ]
countryName             = match
stateOrProvinceName     = match
organizationName        = match
organizationalUnitName  = optional
commonName              = supplied
emailAddress            = optional

# For the 'anything' policy
# At this point in time, you must list all acceptable 'object'
# types.
[ policy_anything ]
countryName             = optional
stateOrProvinceName     = optional
localityName            = optional
organizationName        = optional
organizationalUnitName  = optional
commonName              = supplied
emailAddress            = optional

################################################################
####
[ req ]
default_bits            = 1024
default_keyfile         = privkey.pem
distinguished_name      = req_distinguished_name
attributes              = req_attributes
x509_extensions = v3_ca # The extentions to add to the self
signed cert

# Passwords for private keys if not present they will be
prompted for
# input_password = secret
# output_password = secret

# This sets a mask for permitted string types. There are several
options.
# default: PrintableString, T61String, BMPString.
# pkix   : PrintableString, BMPString.
```

```
# utf8only: only UTF8Strings.
# nombstr : PrintableString, T61String (no BMPStrings or
UTF8Strings).
# MASK:XXXX a literal mask value.
# WARNING: current versions of Netscape crash on BMPStrings or
UTF8Strings
# so use this option with caution!
string_mask = nombstr

# req_extensions = v3_req # The extensions to add to a
certificate request

[ req_distinguished_name ]
countryName                     = Country Name (2 letter code)
countryName_default             = AU
countryName_min                 = 2
countryName_max                 = 2

stateOrProvinceName             = State or Province Name (full
name)
stateOrProvinceName_default     = Some-State

localityName                    = Locality Name (eg, city)

0.organizationName              = Organization Name (eg,
company)
0.organizationName_default      = Internet Widgits Pty Ltd

# we can do this but it is not needed normally :-)
#1.organizationName             = Second Organization Name (eg,
company)
#1.organizationName_default     = World Wide Web Pty Ltd

organizationalUnitName          = Organizational Unit Name (eg,
section)
#organizationalUnitName_default =

commonName                      = Common Name (eg, YOUR name)
commonName_max                  = 64

emailAddress                    = Email Address
emailAddress_max                = 64

# SET-ex3                       = SET extension number 3

[ req_attributes ]
challengePassword               = A challenge password
```

```
challengePassword_min           = 4
challengePassword_max           = 20

unstructuredName                = An optional company name

[ usr_cert ]

# These extensions are added when 'ca' signs a request.

# This goes against PKIX guidelines but some CAs do it and some
software
# requires this to avoid interpreting an end user certificate as
a CA.

basicConstraints=CA:FALSE

# Here are some examples of the usage of nsCertType. If it is
omitted
# the certificate can be used for anything *except* object
signing.

# This is OK for an SSL server.
# nsCertType                    = server

# For an object signing certificate this would be used.
# nsCertType = objsign

# For normal client use this is typical
# nsCertType = client, email

# and for everything including object signing:
# nsCertType = client, email, objsign

# This is typical in keyUsage for a client certificate.
# keyUsage = nonRepudiation, digitalSignature, keyEncipherment

# This will be displayed in Netscape's comment listbox.
nsComment                       = "OpenSSL Generated
Certificate"

# PKIX recommendations harmless if included in all certificates.
subjectKeyIdentifier=hash
authorityKeyIdentifier=keyid,issuer

# This stuff is for subjectAltName and issuerAltname.
# Import the email address.
# subjectAltName=email:copy
```

```
# An alternative to produce certificates that aren't
# deprecated according to PKIX.
# subjectAltName=email:move

# Copy subject details
# issuerAltName=issuer:copy

#nsCaRevocationUrl              = http://www.domain.dom/ca-
crl.pem
#nsBaseUrl
#nsRevocationUrl
#nsRenewalUrl
#nsCaPolicyUrl
#nsSslServerName

[ v3_req ]

# Extensions to add to a certificate request

basicConstraints = CA:FALSE
keyUsage = nonRepudiation, digitalSignature, keyEncipherment

[ v3_ca ]


# Extensions for a typical CA


# PKIX recommendation.

subjectKeyIdentifier=hash

authorityKeyIdentifier=keyid:always,issuer:always

# This is what PKIX recommends but some broken software chokes
on critical
# extensions.
#basicConstraints = critical,CA:true
# So we do this instead.
basicConstraints = CA:true

# Key usage: this is typical for a CA certificate. However since
it will
# prevent it being used as an test self-signed certificate it is
best
# left out by default.
# keyUsage = cRLSign, keyCertSign
```

```
# Some might want this also
# nsCertType = sslCA, emailCA

# Include email address in subject alt name: another PKIX
recommendation
# subjectAltName=email:copy
# Copy issuer details
# issuerAltName=issuer:copy

# DER hex encoding of an extension: beware experts only!
# obj=DER:02:03
# Where 'obj' is a standard or added object
# You can even override a supported extension:
# basicConstraints= critical, DER:30:03:01:01:FF

[ crl_ext ]

# CRL extensions.
# Only issuerAltName and authorityKeyIdentifier make any sense
in a CRL.

# issuerAltName=issuer:copy
authorityKeyIdentifier=keyid:always,issuer:always

[ proxy_cert_ext ]
# These extensions should be added when creating a proxy
certificate

# This goes against PKIX guidelines but some CAs do it and some
software
# requires this to avoid interpreting an end user certificate as
a CA.

basicConstraints=CA:FALSE

# Here are some examples of the usage of nsCertType. If it is
omitted
# the certificate can be used for anything *except* object
signing.

# This is OK for an SSL server.
# nsCertType                    = server

# For an object signing certificate this would be used.
# nsCertType = objsign
```

```
# For normal client use this is typical
# nsCertType = client, email

# and for everything including object signing:
# nsCertType = client, email, objsign

# This is typical in keyUsage for a client certificate.
# keyUsage = nonRepudiation, digitalSignature, keyEncipherment

# This will be displayed in Netscape's comment listbox.
nsComment                       = "OpenSSL Generated
Certificate"

# PKIX recommendations harmless if included in all certificates.
subjectKeyIdentifier=hash
authorityKeyIdentifier=keyid,issuer:always

# This stuff is for subjectAltName and issuerAltname.
# Import the email address.
# subjectAltName=email:copy
# An alternative to produce certificates that aren't
# deprecated according to PKIX.
# subjectAltName=email:move

# Copy subject details
# issuerAltName=issuer:copy

#nsCaRevocationUrl              = http://www.domain.dom/ca-
crl.pem
#nsBaseUrl
#nsRevocationUrl
#nsRenewalUrl
#nsCaPolicyUrl
#nsSslServerName

# This really needs to be in place for it to be a proxy
certificate.
proxyCertInfo=critical,language:id-ppl-
anyLanguage,pathlen:3,policy:foo
```