The OPC UA Security Model For Administrators

Whitepaper Version 1.00

July 7, 2010

Randy Armstrong, OPC Foundation

Paul Hunkar, Yokogawa

1 Introduction

1.1 Background

A security model is an architecture that allows developers, administrators and end users to use applications in distributed environment while ensuring that the applications, the computers they run on and the information exchanged is not compromised. A complete security model has several facets including application security, transport security, user authorization and authentication and traceability. This white paper describes how to use the OPC UA security model to ensure application and transport security. The target audience for this document are systems administrators and end users. A second whitepaper will discuss the security model from the perspective of a software developer.

The OPC UA security model has been designed to meet the requirements of many different systems while using the same infrastructure. In order to accommodate different security and administrative requirements the OPC UA security model offers four tiers for application authentication and two tiers for certificate management. It is up to the administrator to decide which tiers best match their needs. Applications should support all tiers. This document also discusses the administrative procedures required by a tier. Applications must allow administrators to configure the level of security enforced by their application just like web browsers allow administrators to configure the security level enforced by the browser.

For this white paper it is assumed that the reader has some basic understanding of OPC UA. The paper will list references that provide a more detailed description of some key concepts utilized by OPC UA with regard to security. A user deploying an OPC UA application may desire to have a deeper understanding of the security concepts, but it is NOT required. OPC UA Applications have all of the required security features built in, many of which are provided by communication stacks and tool kits minimizing the security related work required by developers. The OPC Foundation also provides some simple tools for assisting users with security related tasks. This paper will also provide a description of some of the available tools to help with the administrative procedures required for security management

1.2 Context

The UA Security Model defines four principal actors: the Application Instance, the Application Administrator, the Application Operator and the Certificate Authority. The relationships between these actors are shown in the following figure. Each of the entities is described in the text that follows.

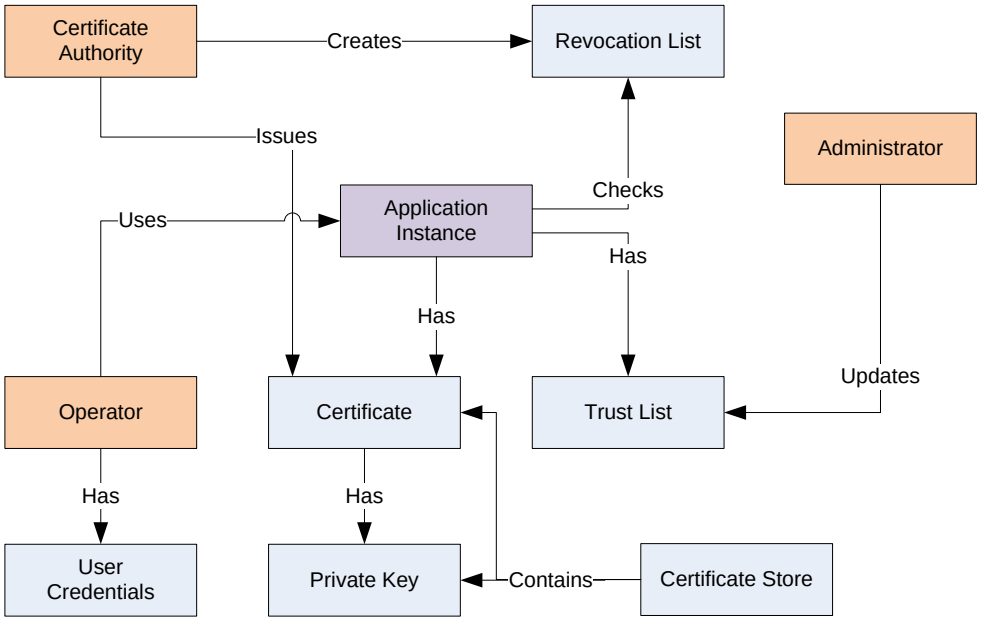


Figure 1 - Security Term and Interactions

Application Instance – An OPC UA Application installed on a single machine is called an Application Instance. Each instance must have its own Certificate which it uses to identify itself when connecting to other applications. Each Application Instance has a globally unique URI which identifies it.

Administrator - An Application Instance must have an Administrator which manages the security settings for the Application.

Operator – An Operator is person who uses the Application Instance. More than one Operator may exist for any given application. An Operator may have User Credentials which are used to determine access rights and to track activities.

Certificate Authority (CA) - A Certificate Authority (CA) is an administrator or organization which is responsible for creating Certificates. The Certificate Authority must verify that information placed in the Certificate is correct and add a digital signature to the Certificate that is used to verify that the information has not been changed. Each CA must have its own Certificate which is used to create the digital signatures.

Certificate - A Certificate is an electronic ID that can be held by an application. The ID includes information that identifies the holder, the issuer and a unique key that is used to create and verify digital signatures. The syntax of these certificates conforms to the X509 specification, as a result, these certificates are also called “X509 Certificates”. Certificates also have a Private Key associated with them.

User Credential – A User Credential is a generic term for an electronic ID which identifies an Operator. It may be passed to a Server after the Application Certificate is used to create a secure channel. It is used to determine access rights and to track activities.

Private Key - A Private Key is a secret number known only to the holder of a Certificate. This secret allows the holder to create digital signatures and decrypt data. If this secret is revealed to unauthorized parties then the associated Certificate can no longer be trusted or used.

Certificate Store - A Certificate Store is a place where Certificates and Private Keys can be stored on a file system. All Windows systems provide a registry based store called the Windows Certificate Store. All systems support a directory containing the Certificates stored in a file which is also called an OpenSSL Certificate Store.

Self-Signed Certificate - A Self-Signed Certificate is a Certificate which has no Certificate Authority. These Certificates can be created by anyone and can be used in situations where the administrators of UA Applications are able to verify the claims by reviewing the contents themselves and security is addressed in another manner.

Trust List - A Trust List is a list of Certificates which are trusted by an Application Instance. When security is enabled UA Applications must reject connections from peers if they do not have a Certificate that is in the trusted or issued by a CA that is in the Trust List.

Revocation List - A Revocation List is a list of Certificates which have been revoked by a CA and must not be accepted by an Application Instance.

2 Security Tiers

2.1 The Basics

In OPC UA, each installation of an application must have an application instance certificate that uniquely identifies the application and the machine that it is running on. These certificates come with private keys that allow applications to create secure communication channels that cannot be viewed by 3rd parties or modified while in transit. These certificates also allow OPC UA applications to be identified by peers and to block communication from a peer if it is not authorized.

2.2 Tier 1 - No Authentication

In this tier the client and server allow any peer to communicate which means that all valid certificates are trusted. The application certificates are used only to provide unverifiable information about the peer. The receiver has no way to know if the sender is the legitimate holder of the certificate.

In this mode the client and server automatically accept valid certificates even if they have not been explicitly added to the trust lists managed by the client and server applications. This mode requires no configuration at the server or client.

This tier cannot ensure the privacy of any information transmitted, including user credentials. This tier would only be appropriate in a system that has guaranteed security in some other manner, such as a physically secured and isolated system or where communications is secured via VPN or other such transport layer security. It would also be appropriate in a situation where all information is public and access to it is open.

A developer need only configure an installation procedure that generates an application instance certificate for an application on installation.

2.3 Tier 2 - Server Authentication

In this tier the server allows any client to connect and if user authentication is required it is done by sending user credentials such as a username/password after the secure channel has been established. Clients, on the other hand, must be configured by the administrator to trust the Server.

Clients will trust a Server if an administrator has explicitly placed the Server certificate into its trust list or if the Server's certificate was issued by a CA which is in its trust list.

If the Server's certificate was not explicitly in the trust list (i.e. the certificate was issued by a CA that is in the trust list) the client should compare the DNS name in the Server's certificate to the DNS name it used to connect to server. If they match the client knows it is connecting to the machine it thinks it is connecting to. This does not guarantee that the client has connected to correct server, only that the machine is the correct machine.

This tier is used by most Internet banking applications where the bank's web server has a certificate issued by Certificate Authorities like Verisign which are automatically placed in the browsers trust list by the Windows operating system. It provides fairly good security but the server cannot restrict the client applications.

2.4 Tier 3 - Client Authentication

In this tier the client connects to any server but the server only allows trusted clients to connect. Clients never provide sensitive information since it does not know if it the server is legitimate.

In this tier clients need no pre-configuration other than the URL of the server. However, Servers will only trust clients with certificates that have been placed by administrators in the servers trust list or if the Clients certificate was issued by a CA which is in its trust list

This mode is used by discovery services which need to ensure that only authorized applications have access to them but clients don't care if the server is not legitimate. The local discovery server (LDS) operates in this mode and only allows authorized applications to register themselves.

2.5 Tier 4 - Mutual Authentication

In this tier both the client and server only allow trusted peers to connect. It offers the highest level of security but requires that both the client and server be configured in advance. This is the recommended mode for any public or semi-public deployment of OPC UA or for deployments where security is a primary concern.

As in Tier 2, clients should check the DNS name if the Server certificate was not explicitly placed in the trust list.

It will be used in environments where administrators want complete control over which applications can talk to each other. It also provides the most secure environment.

Application installation should default to Tier 4 mode.

3 Certificates and Certificate Stores

3.1 Overview

When working with certificates it is important to understand the formats associated with a certificate and where the certificates are stored. Both the formats and storage location vary from platform to platform and also vary by the cryptographic library that is being used by an application.

3.2 Certificates and Private Keys

Certificates are typically stored in files that can have a number of formats.

The formats used by UA applications are shown in the following tables.

DER	An ASN.1 blob encoded with the DER (distinguished encoding rules). File extension is `*.der` or `*.cer` on Windows systems. Use only for storing the Certificate (not the Private Key). Certificates can be imported or exported to/from Windows Certificate Stores using this format. It is also the file format used to store a certificate in a directory store.
PKSC#12	A binary format used to store RSA private keys with their certificates. File extension is `*.pfx`. May be password protected. Private keys can be imported or exported to/from Windows Certificate Stores using this format.
PEM	A text format used to store private keys File extension is `*.pem`. May be password protected. This format is only used by some OpenSSL based applications or windows application, but they include items such as CRL lists.

Other formats such as `*.crt`, or `*.crl` may occur in some systems, but all others can be converted or matched to one of the above, by the operating system

3.3 Windows Certificate Stores

Windows Certificate Stores are accessible via standard Windows tools and WIN32 APIs. They are physically stored in the registry but must be accessed via the standard APIs.

There are two special store locations: LocalMachine and CurrentUser:

- The LocalMachine location contains Certificate stores shared by all users and services on a machine. All users have read access to these stores.
- The CurrentUser (or CurrentService) location contains Certificate stores which are only accessible to the current user or service. These stores can be accessed by administrators via the standard APIs.

The Windows Certificate Stores are identified by the location and a store name separated by a backslash. e.g. `LocalMachine\UA Applications or CurrentUser\UA Applications`.

Certificates placed in Windows Certificate Stores may have private keys associated with them; however, users will not be able to access these private keys unless they have been granted permission. The UA Certificate Tool can be used to manage permissions for private keys.

3.4 Directory Stores

Any directory can contain `*.der`, `*.pfx` or `*.pem` files. These directories are called a certificate store and identified by the full path of the directory.

By convention the UA Stacks also support a directory store which places the private keys in a separate subdirectory. The subdirectory for certificates is called ‘certs’ and the subdirectory for private keys is called ‘private’. This style of certificate store is also called an OpenSSL store.

An OpenSSL store is identified by the full path of the root directory. The presence of the ‘certs’ subdirectory is used to distinguish between a simple directory store and an OpenSSL store.

OpenSSL may also make use of directories such as `crl`, which would contain certificate revocation lists. This directory is at the same level as the `certs` directory

Some examples create CA certificates at the parent folder of `certs` directory (its private key in the private directory) with certificates signed by CA key in the `certs` directory. For OPC UA a CA directory is created, see Figure 2 for an illustration

Figure 2 - Directory Stores

3.5 Key Certificate Properties

A certificate contains a number of fields that have functionality as defined in the x.509 specification. A few key fields are listed here:

Common Name:	This is usually the Application Name obtained from the configuration file associated with the application for which the certificate is being generated, but it must be an appropriate name for the application.

Organization:	This is usually a description of the organization where the application is installed.
SubjectAltName: URI:	This is the unique Application URI associated with the application, and is obtained from the configuration file associated with the application for which the certificate is being generated. Only one URI field may be in a certificate.
SubjectAltName: DNSName IPAddress	The DNS name or IP Address of the machine where this application is installed. Multiple DNS Names and/or IP Addresses can be in a single certificate.
Valid from **Valid to**:	This is the starting and ending date for which a certificate is valid. The Automatic certificate checking will verify that the current date is between these dates when validating a certificate. The administrator should check these dates periodically and replace any certificates that are about to expire with new certificates.

These fields are usually used by an administrator to help identify and match a certificate to an application. OPC UA applications also automatically check some of these fields when verifying a received certificate from an application. Certificates contain additional information, but this information is usually provided by the tool that generates the certificate.

4 Tools

4.1 Overview

This section will describe some of the common tools that can be used to manage OPC UA Applications and certificates. The tools include:

- **UA Configuration Tool** – Manages OPC UA Applications, Security Settings, Certificates and more. A general purpose tool provided by the OPC Foundation that simplifies OPC UA management.
- **UA Certificate Generator** – Generate Application instance certificates. These Certificates can be CA based or self-signed. This tool is written in C++ and uses OpenSSL thus can be adapted to almost any platform
- **Microsoft Management Console** – The Microsoft provided Certificate manager.
- **OpenSSL**- An open source library that is already ported to many platforms and provides cryptographic functions along with a command-line tool for generating and managing certificates.

4.2 UA Configuration Tool

4.2.1 Overview

The UA Configuration Tool is GUI application that allows Administrators to do the following:

1) Manage Security
2) Manage Applications
3) Manage Certificates
4) Manage COM Interoperability
5) Manage HTTP Access Rules

The UA Configuration Tool is included in the Local Discovery Server (LDS) Merge Module and is installed by any package that installs the LDS. When it is installed it can be found in the following directory: `$(CommonProgramFiles)\OPC Foundation\UA\v1.0\Bin`

Where `$(CommonProgramFiles)` is the location of the `$(CommonProgramFiles)` directory on the system. The x86 version of the directory is used if the system is running a 64-bit operating system.

A link is also placed in the Start Menu under "All Programs | OPC Foundation | UA SDK 1.0"

4.2.2 Choosing an Application to Manage

The UA Configuration Tool has a drop down that is shared by multiple tabs allowing the Administrator to select an Application to manage. This drop down is populated with a list of applications installed on the machine from a set of configuration files placed in the `$(MyDocuments)\OPC Foundation\Applications` directory.

The files are XML files that can be generated by the Configuration Tool or by the Applications themselves. Vendors that wish to facilitate configuration of their applications can create these files and place them in the directory where they will be found by the tool. The tool can also be pointed to a file that is in another location using the find button.

The XML file has the following layout:

```
<ManagedApplication xmlns="http://opcfoundation.org/UA/SDK/Configuration.xsd">
  <DisplayName>Opc.Ua.ComProxyServer</DisplayName>
  <ExecutablePath>
    X:\OPC\UA313\Source\Bin\Opc.Ua.ComProxyServer.exe
  </ExecutablePath>
  <ConfigurationPath>
    X:\OPC\UA313\Source\Bin\Opc.Ua.ComProxyServer.Config.xml
  </ConfigurationPath>
  <Certificate>
    <StoreType>Directory</StoreType>
    <StorePath>
      %CommonApplicationData%\OPC Foundation\CertificateStores\MachineDefault
    </StorePath>
    <SubjectName>UA COM Proxy Server</SubjectName>
  </Certificate>
  <TrustList>
    <StoreType>Windows</StoreType>
    <StorePath>LocalMachine\UA Applications</StorePath>
  </TrustList>
</ManagedApplication>
```

Each of the fields is described in the following table:

DisplayName	A human readable name for the application. This is displayed in the drop down list.
ExecutablePath	The full path to the executable file.
ConfigurationFile	The full path to the configuration file. The tool is able to modify the file if it is compatible with the SecuredApplication schema defined in Part 6 - Annex D. If the tool cannot parse the file it can still be used to control access to the file. All applications built with the UA .NET SDK have a compatible configuration schema.
Certificate	The location of the application instance certificate.

StoreType	The type of certificate store. Must be Windows or Directory
StorePath	The location of the certificate store.
SubjectName	The CommonName or full SubjectName in the certificate. The CommonName is the portion of the SubjectName prefixed with 'CN='.
TrustList	The location of the application trust list.

A new managed application can be created within the Configuration Tool by clicking on the 'Find...' button and navigating to the location of the executable file. At that point the following dialog will be displayed:

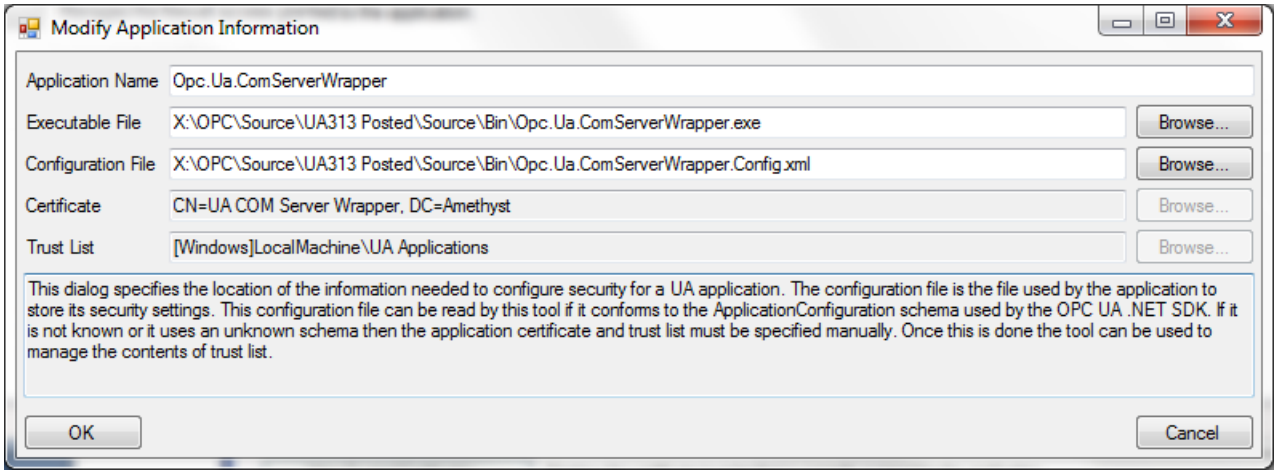


Figure 3 - Modify Application Information

The tool will attempt to detect the configuration file and see if it is compatible with the Secured Application schema. If it is then the remaining fields will be filled-in. If it is not then the remaining fields must be filled-in manually. This process will require some knowledge of the application being configured. It is recommended that Application developers provide Application configuration files when they deploy OPC UA Applications

4.2.3 Manage Security

This tab allows an administrator to review and manage the security settings associated with the selected application instance (see Figure 4). The GUI allows for easy selection of the available OPC UA applications.

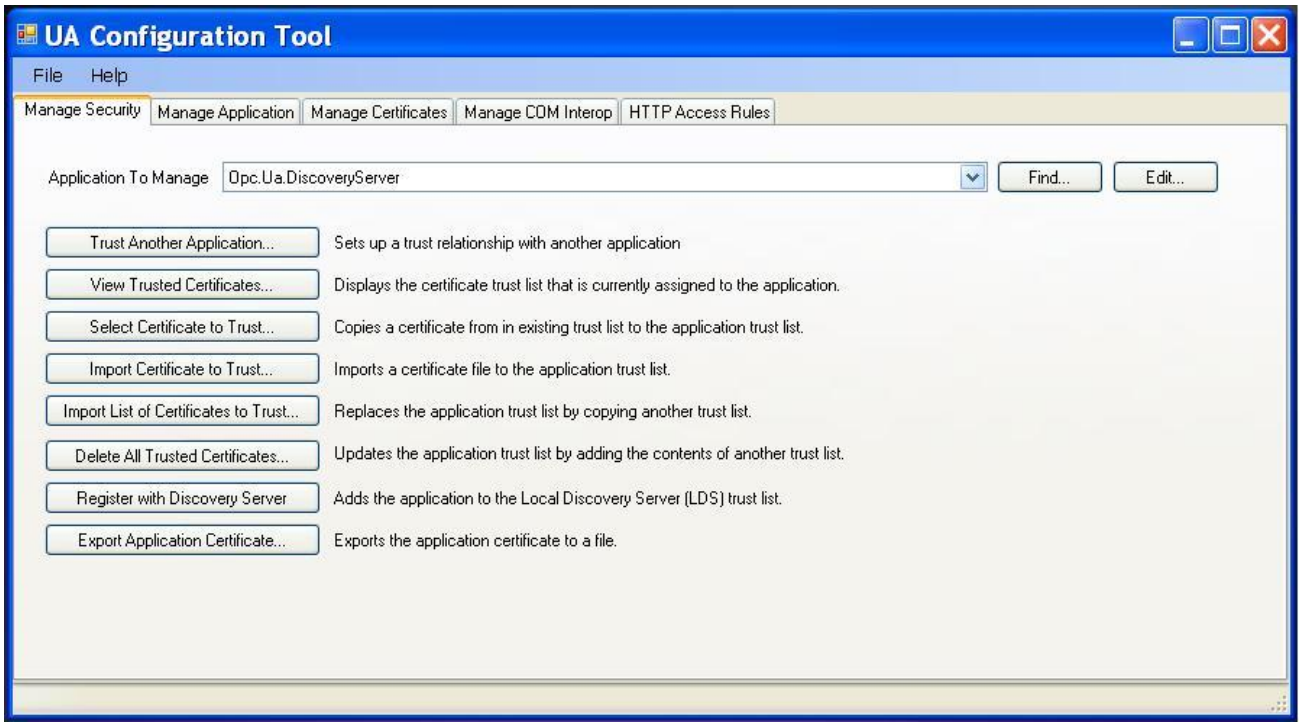


Figure 4 - UA Configuration Tool - Manage Security

The following configuration options are available for the Application that is selected in the common drop down:

- Trust Another Application
- View Trusted Certificates
- Select Certificate to Trust
- Import Certificate to Trust
- Import List of Certificates to Trust
- Delete all Trusted Certificates
- Register with Discovery Server
- Export Application Certificate

4.2.3.1 Trust Another Application

All Applications need to be configured with list of other applications that they trust. This option allows an administrator to select another application from the list of applications, which results in the selected application's Certificate being added to the list of trusted certificates for application that is being configured.

4.2.3.2 View Trusted Certificates

This option displays the list of certificate that the application trusts (see Figure 5). This list includes any Certificate Authority (CA) certificates. The list can be filtered in multiple ways, allowing for easy review. Administrator can use this list to check the expiration dates of certificates and renew any certificate prior to their expiration. The administrator can also use the list to ensure that only application that are authorized applications are in the trust list.

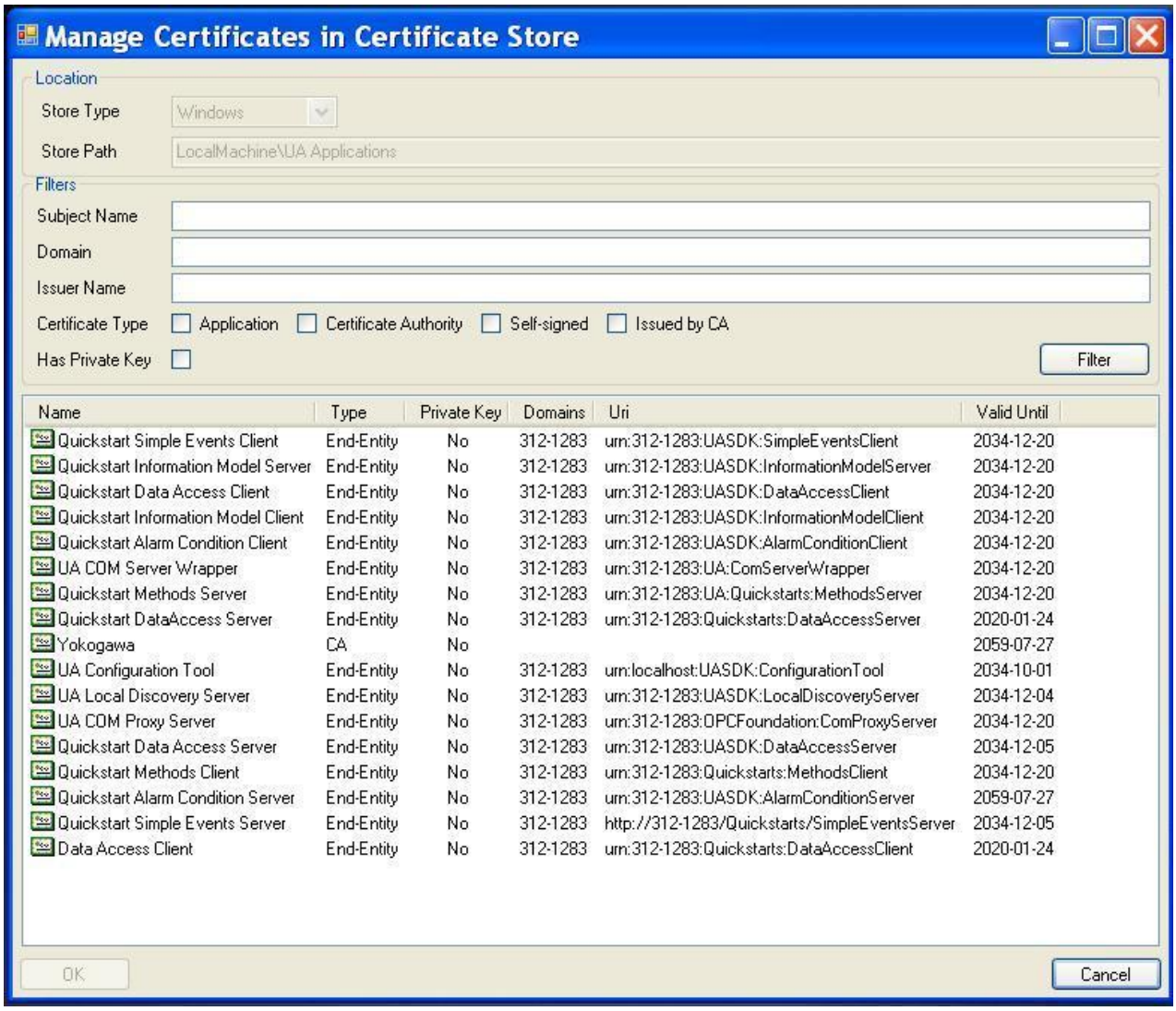


Figure 5 - UA Configuration Tool - Certificate Store

4.2.3.3 Select Certificate to Trust

This option allows an administrator to browse any of the available certificate storage locations, including both windows stores and directory stores and select certificates that the given application should trust. The administrator should review the certificate in detail before selecting it for addition to the trust list. It is the administrator responsibility to ensure that the certificate correctly reflects an application that should be trusted.

4.2.3.4 Import Certificate to Trust

This option allows an administrator to select a certificate and add it to the list of trusted certificates. The certificate can be on any location including flash drives or network share locations. It is the administrator's responsibility to review the certificate and ensure that it is a certificate that belongs to an application that is to be trusted.

This function verifies the signature on the certificate that is being imported. To verify a signature, the Configuration Tool must have all of the CAs associated with the certificate in its trust list. For that reason, the trust list for the Configuration Tool defaults to LocalMachine\UA Certificate Authorities instead of LocalMachine\UA Applications. This allows administrators to import CA certificates for verification purposes without affecting other applications.

4.2.3.5 Import List of Certificates to Trust

This option allows an administrator to import an entire list of certificates or to move a list from one store to another. This feature can be used along with a network share to build a list of certificates that are to be stored and then import the list for a new installation. Care should be taken that the list of certificates is not tampered with and all are the expected certificates.

4.2.3.6 Delete all Trusted Certificates

This option allows an administrator to clean up the certificate list quickly. This feature would be used most commonly at the conclusion of a system test or other such period, just prior to a deployment at a job site i.e. where all of the listed certificates were intended for use during testing, but are not appropriate for the job site. It may also be used at a job site if a machine is being move from one area to another and the trust list needs to be reset.

4.2.3.7 Register with Discovery Server

This option allows an administrator to register the selected application's certificate with the Discovery server as a trusted certificate.

4.2.3.8 Export Application Certificate

This option allows an administrator to generate a file that contains the certificate. This file can them be used to import the certificate onto another machine. The resulting file is a .DER file and does **not** contain the Private Key information.

4.3 Manage Application

This tab allows an administrator to review and manage the security settings associated with the selected application instance. The GUI allows for easy selection of the available UA applications.

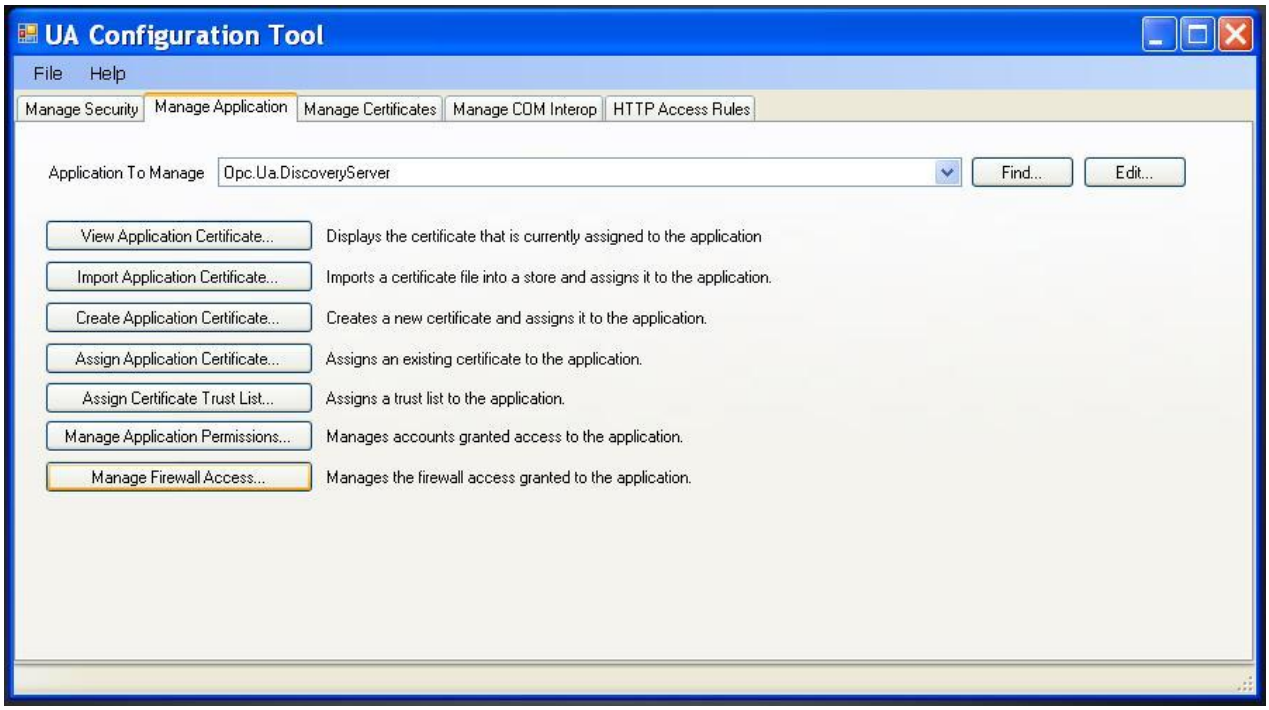


Figure 6 - Configuration Tool - Manage Application

The following configuration options are available:

- View Application Certificate
- Import Application Certificate
- Create Application Certificate
- Assign Application Certificate
- Assign Certificate Trust List
- Manage Application Permissions
- Manage Firewall Access

4.3.1.1 View Application Certificate

This option displays the current application certificate in the following dialog:

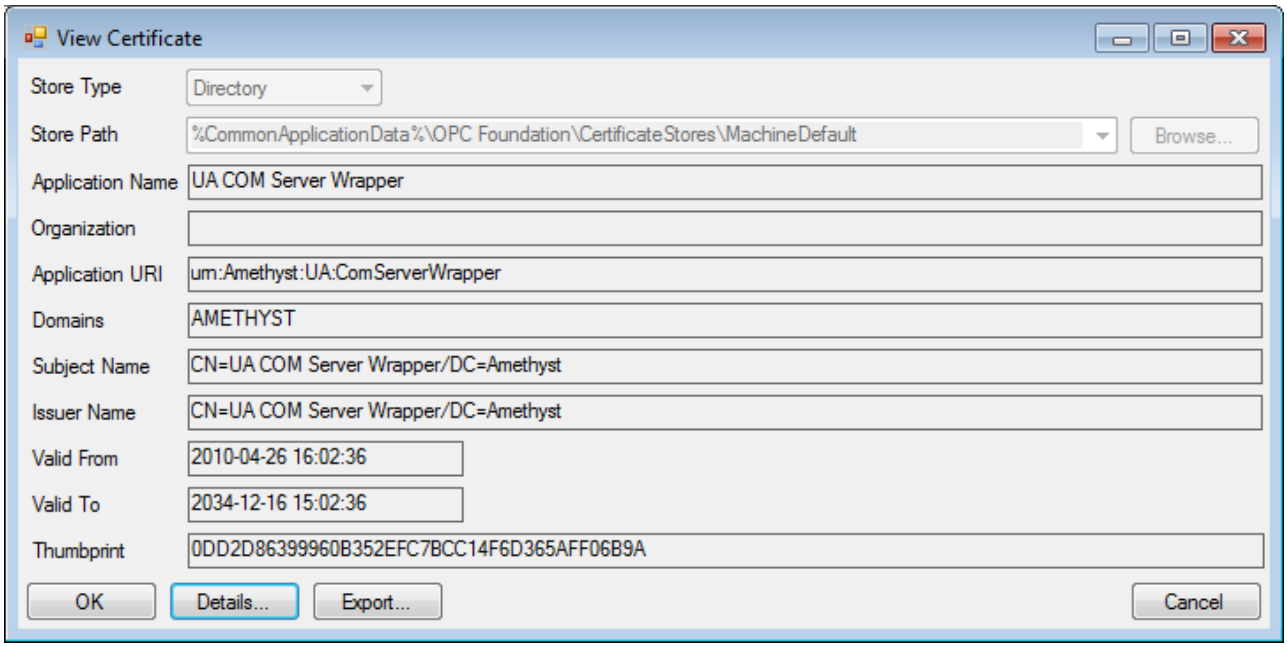


Figure 7 - UA Configuration Tool – View Application Certificate

The Export option saves the Certificate to a .DER file. This save does **not** include the Private Key.

4.3.1.2 Import Application Certificate

This option imports an application certificate from a .PFX file stored on disk. It prompts the user to enter a password if one is required. The imported Certificate will replace any Application Instance certificate that may already be assigned to the application.

4.3.1.3 Create Application Certificate

This option creates a new application certificate with the following dialog:

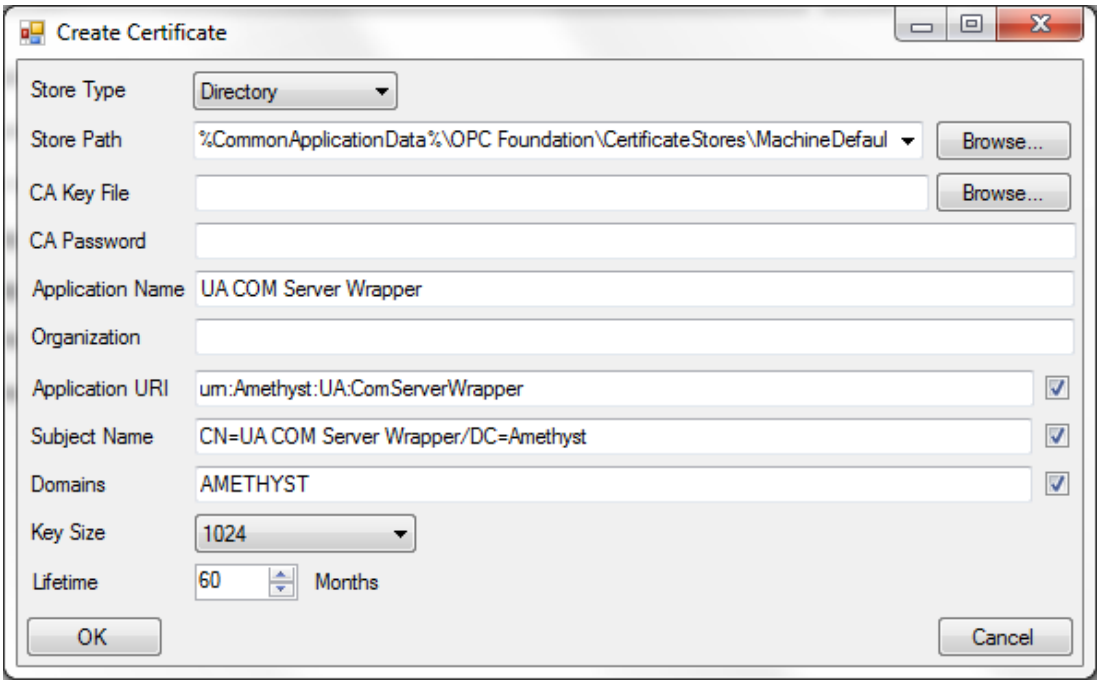


Figure 8 - UA Configuration Tool – Create Certificate

The CA Key File is a PFX file containing the Certificate Authority private key. If left blank a self-signed certificate is created. If provided the CA Password may also be required. If the Application URI/Subject Name/Domains checkboxes are unchecked the tool will generate them automatically from the other information provided.

The Store Path specifies where the application certificate will be place after it is created.

4.3.1.4 Assign Application Certificate

This option opens the certificate store dialog shown in Figure 5. Allows the administrator to select a certificate and assign it to the application.

4.3.1.5 Assign Certificate Trust List

This option prompts the administrator to select a certificate store to use as a trust list.

4.3.1.6 Manage Application Permissions

This option allows the administrator to manager access to the application. When this option is picked the dialog in Figure 9 appears.

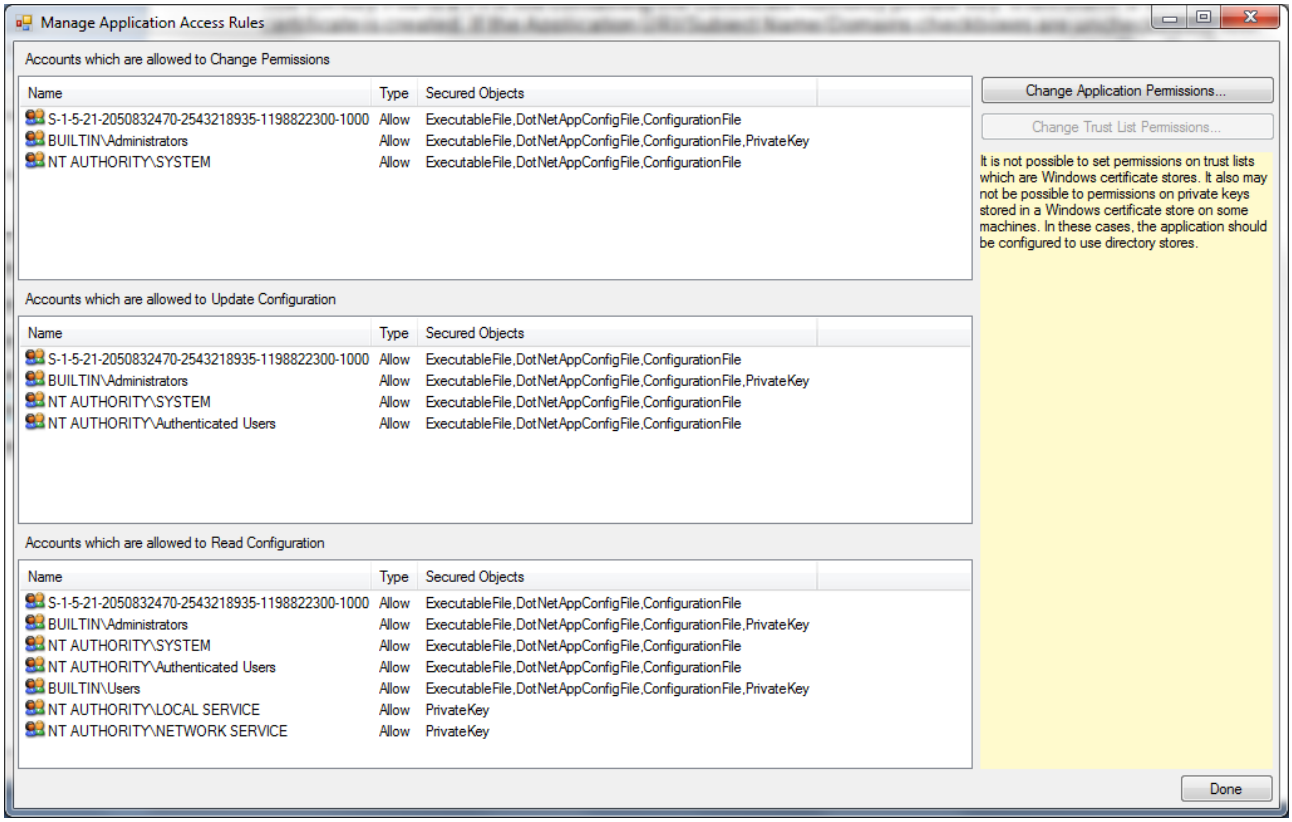


Figure 9 - UA Configuration Tool – Manage Application Permissions

The dialog looks at the current permissions for all of the objects which affect application security. Accounts with the ability to change permissions are able to grant access to other users so these accounts should be limited.

These objects are summarized in the following table:

ExecutableFile	This is the executable file for the application. All operators that are allowed to launch the application must have read access.

DotNetAppConfigFile	This is the .NET app.config file. It is only present for .NET based applications. Only Administrators can have update access. Operators must have read access.
ConfigurationFile	This is the configuration file which contains the security settings. Only Administrators can have update access. Operators must have read access.
PrivateKey	The private key associated with the application instance certificate. Only Administrators can have update access. Operators must have read access. Some administrators may not want to allow operators to have any access to the private key. This requirement can be accommodated by only granting administrators access to the private key and having an administrator launch the application. After that the operator would be limited to what application allowed it to do.
TrustList	The trust list associated with the application. Trust lists can be shared so permissions may be granted to different users. Windows stores do not allow permissions to be configured.

4.3.1.7 Manage Firewall Access

This option allows the administrator to manage the firewall access granted to the application.

4.3.2 Manage Certificates

This tab allows an administrator to review and manage the certificates on a machine. This includes the ability to act as a Certificate Authority (CA). The administrator must provide the key file for a CA certificate, it acting as a CA.

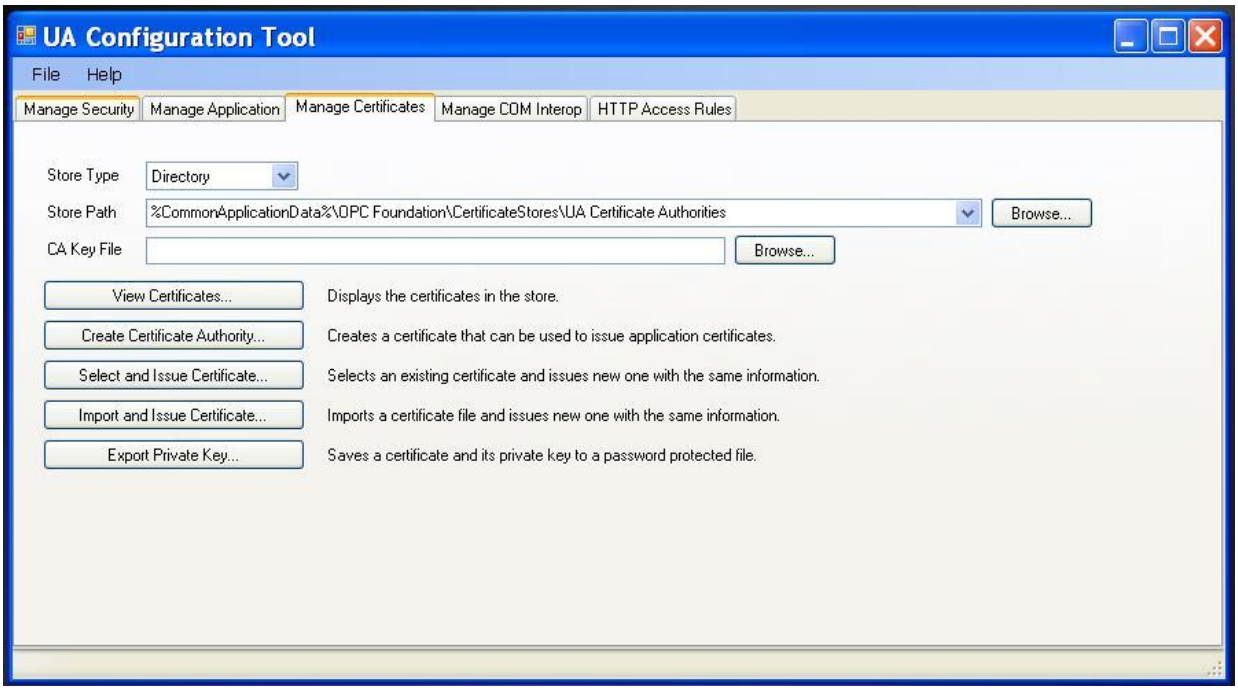


Figure 10 - Configuration Tool - Manage Certificates

This tab provides access to the following options:

- View Certificates,
- Create a Certificate Authority Certificate,
- Select and Issues Certificates
- Import and Issue a Certificate
- Export a Private Key.

4.3.2.1 View Certificates

This option views the certificates in the currently selected certificate store. It allows an administrator to view, filter, manage and even delete certificates in multiple certificates stores.

4.3.2.2 Create a Certificate Authority Certificate

This option creates a certificate authority.

Note that the private key for a CA certificate must be carefully protected. It is strongly recommend that a password or a directory with restricted access be used.

The CA certificate can be a sub-CA or a standalone CA. This allows the administrator to create a tree of Certificates Authorities, which may be useful in a large industrial setting where each portion of the plant may have multiple machines and UA applications and trusts need to be established between all of the machines, but the various portions of the plant should be kept separate, with only a select few administrative UA Applications with access to all aspects of the plant.

4.3.2.3 Select and Issues Certificates

This option greatly simplifies the issuing of CA based certificates for an administrator. Since all applications generate a self-signed certificate on installation, the administrator can select the self-signed certificate for the application and then generate CA certificate for the same application without having to enter any information. If the Administrator is not using a CA, then the administrator can still issue updated certificates to replace expiring certificates or to replace a compromised certificate very quickly and easily.

4.3.2.4 Import and Issue a Certificate

This option provides the Administrator the same functionality as described in section 4.3.2.3 except that the certificate can reside on some external device or network share.

4.3.2.5 Export a Private Key.

This option allows an administrator to generate certificate on a central computer and to export it as a file that can be import on alternate machines. Care must be taken with any exported certificate, since the private key is included and could be used by any application that gains access to it. The private key should be password protected or it should be placed on some secure storage device.

4.3.3 Manage HTTP Access Rules

A user must be granted permission to launch any OPC UA Application which exposes an HTTP endpoint. Administrators can manage the rules on the current machine via the HTTP Access Rules tab as shown here:

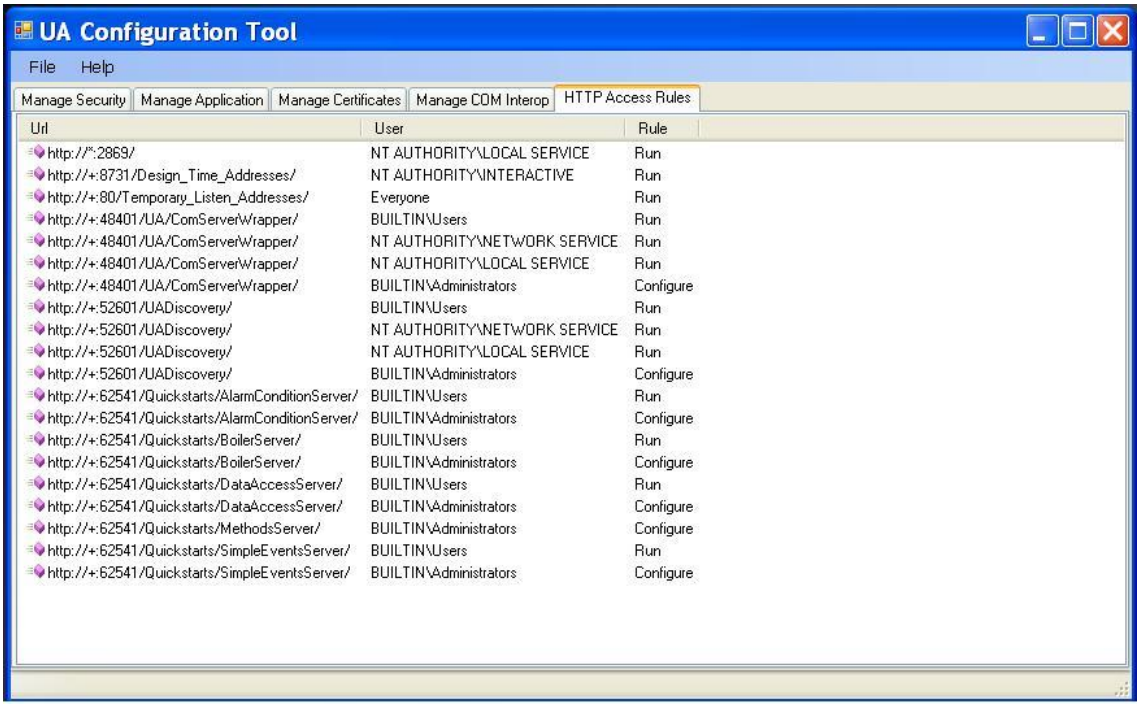


Figure 11 - Configuration Tool – HTTP Access Rules

Rules can be deleted by selecting the rule and using the right mouse button to bring up a context menu. At the time of this writing, the only action supported by the tool is deletion. In the future additional option may be available.

4.4 UA Certificate Generator

The UA Certificate Generator is a command line tool that allows Administrators to do the following:

1) Create new Self-Signed Certificate
2) Create new CA certificate
3) Create new CA signed certificate
4) Revoke a Certificate (future)
5) Generate a CRL list (future)

The UA Certificate Generator is included in the Local Discovery Server (LDS) Merge Module and is installed by any package that installs the LDS. When it is installed it can be found in the following directory: `$(CommonProgramFiles)\OPC Foundation\UA\v1.0\Bin`

Where `$(CommonProgramFiles)` is the location of the `$(CommonProgramFiles)` directory on the system. The x86 version of the directory is used if the system is running a 64-bit operating system. This tool can is also available on non-windows platforms.

The tool supports the following command line parameters

storePath	The location of the directory store where the certificate will be placed. If the directory does not exist then it is created automatically.
applicationName	The name of the application. This must be provided.
applicationUri	A globally unique URI for the application. If not provided the default is constructed from the machine name and the applicationName. e.g. `urn:machinename:applicationName`
organization	The organization name of the certificate. If not provided, no organization is listed.
subjectName	The subject name of the certificate. This is a sequence of X500 name-value pairs. e.g. `CN=MyApplication,O=MyCompany` Each pair is separated by a comma. Values cannot contain commas. The Common Name (CN) should be the applicationName If not provided, a default value is constructed from the machine name, the applicationName and the organization name. e.g. `CN=applicationName,DC=machineName and O=organizationName`
domainNames	This is a sequence of DNS names that are added to the certificate. Each name is separated by a comma. If not provided the default is the machine name.
keySize	This is the length of the key in bits. It may be 1024 or 2048. 1024 is the default.
lifetimeInMonths	This is the lifetime of the certificate in months. The default is 60.
ca	The parameter value can only be true or false. Indicates that a CA certificate is to be generated.
password	The password to be associated with the new generated certificate. If omitted than the certificate is not password protected.

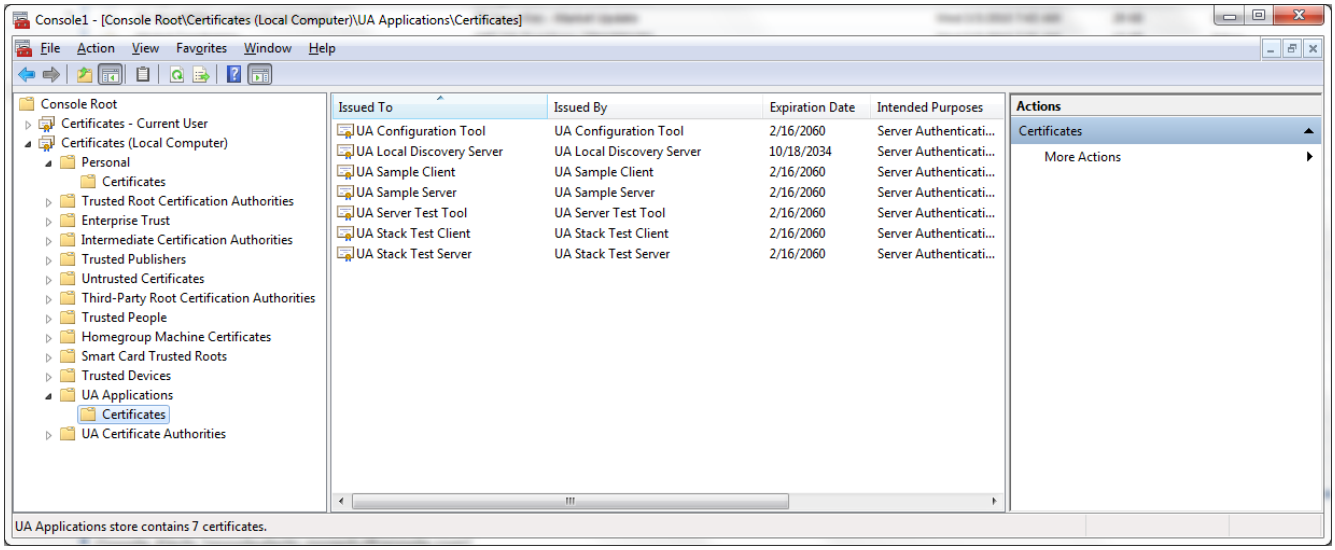
issuerKeyFilePath	The path to a CA certificate that is to be used for generating the certificate. If not provided then a self signed certificate is produced.
issuerKeyPassword	The password for the CA Certificate provided in issuerKeyFilePath

If no command-line parameters are provided then the tool reads them from STDIN. When reading parameters from STDIN a carriage return (\n) is used to separate each parameter. A blank line is used to indicate that all of the parameters have been provided. This command-line tool makes use of the OpenSSL library. It simplifies the commands that would be required for manual generation when compared with using the OpenSSL directly as a command line tool.

4.5 Microsoft Management Console

The Microsoft Management Console is a GUI application that allows Administrators to do the following:

1) Update Application Trust Lists
2) Manage Certificate Permissions



To start-up certificate management via the Microsoft Management Console (MMC), the user must first configure the add-in for certificate management in the MMC. This can vary between versions of the operating system, see Microsoft for details. For detailed instructions on using this tool please refer to the appropriate Microsoft documentation.

4.6 OpenSSL

OpenSSL provides a wealth of functionality, including certificate generation, Certificate Authority functionality and certificate verification. To make use of this functionality requires knowledge of the package and knowledge related to the needs of an OPC UA Application. OpenSSL also only provides support for directory stores. The following are some sample commands, but each of these commands also requires a customized configuration/script file be pre-configured.

- Create a self-signed certificate (Extra)

  A self-signed certificate is a certificate whose issuer is the same as the common name(CN) on the cert, no need for a CA.

  Generate private key

```
openssl genrsa 1024 > MyApplication.key
```

Generate request

```
openssl req –new –key MyApplication.key –out  MyApplication.csr
```

Sign the certificate

```
openssl req –x509 –key MyApplication.key -in MyApplication.csr –
out MyApplication.crt –days 1000
```

- Create Root Certificate

  This is the cert that you will use to sign all other cert requests. If you receive errors regarding the certificate authority for certs signed by your homegrown CA, you must import this CA cert into the "Trusted Root Certificate" store. [Note: The `–config` parameter points to the `/openssl.cnf` file, which is a custom configuration file that can be found on at the OpenSSL web site]

  ```
  openssl req -new -x509 -extensions v3_ca -keyout
  private/cakey.key -out cacert.crt -days 365 -config ./openssl.cnf
  ```

- Create a private key and certificate and sign with your CA

  Generate private key

  ```
  openssl genrsa 1024 > MyApplication.key
  ```

  Generate request

  ```
  openssl req –new –key MyApplication.key –out MyApplication.csr
  ```

  Sign request

  ```
  openssl x509 -req -days 365 -in MyApplication.csr -CA cacert.crt -
  CAkey private/cakey.key -CAcreateserial -out MyApplication.crt
  ```

- Revoke a certificate

  ```
  openssl ca -config X509CA/openssl.cnf -revoke
  X509CA/certs/CertName.pem
  ```

  (Note this command will prompt for a CA password and is done on the CA)

- Generate the CRL list

  ```
  openssl ca -config X509CA/openssl.cnf -gencrl -out crl/crl.pem
  ```

  (Note this command will prompt for a CA password and is done on the CA)

5 Mapping UA to a tier

5.1 Overview

This section will describe the action a vendor would need to perform for their application to easily support the described tiers and the actions for end users to deploy a well behaved application in the given tiers (all are just recommendations) they will include what is recommended which implies that a CA is deployed for tiers 2-4.

5.2 Actions by the Vendor

In OPC UA the samples and some toolkits automatically provide tier 1 type security. This is accomplished by installation programs that generate a self signed certificate.

For most other tiers vendors should provide guidelines on how to deploy their applications so that certificates signed by a CA can be used by the application.

5.3 Actions by the end user

An End User site may require any of the security tiers to be implemented. These guidelines can assist the End User in understanding what must be accomplished for each tier. Each of the cases provides a typical installation environment; these environments are just examples and in no way imply the only environments.

5.3.1 Case 1 – Windows Domain

Let's assume the system is a group of PCs running any combination of Windows operating systems in a domain environment. The PCs have access to the Domain Controller and the PCs are provided by multiple vendors, each with their own configurations.

Tier 1 – would require no additional effort on the part of an end user. Most vendor provided clients and servers would automatically provide the self signed Certificate that is required for this type of installation.

Tier 2, Tier 3, Tier 4 – All include the same type of effort for an administrator, only the list of certificates would change. It is recommended that a CA be defined in the system either using an OPC Foundation tool or a Windows Certificate Server (appropriately customized). See *Figure 13* for an illustration. If the Windows Certificate Server is used, some additional requirements exist such as a server OS, IIS and other details specified in the Windows documentation. It is recommended that Windows documentation be consulted to ensure that all setup and deployment options are correct. A custom template certificate should also be generated that matches the requirements listed for UA certificates. An easier solution is to use the OPC Foundation tool to manage the CA certificates. The administrator for each application as they are deployed would issue a request to the CA and obtain a certificate from the CA (how this is done depends on the tool being used for a CA). They would also install the CA certificate as a trusted CA and would install any CRL lists. If a CA is not desired then as an alternative the Self-signed certificates can be copied via a manual certificate coping process illustrated in *Figure 12*. The OPC UA Configuration tool can assist with this operation by exporting and importing certificates and easily allowing them to be added to an application's trust list, but the complete list of trusted certificate must be copied to each machine.

Manual Certificate Handling

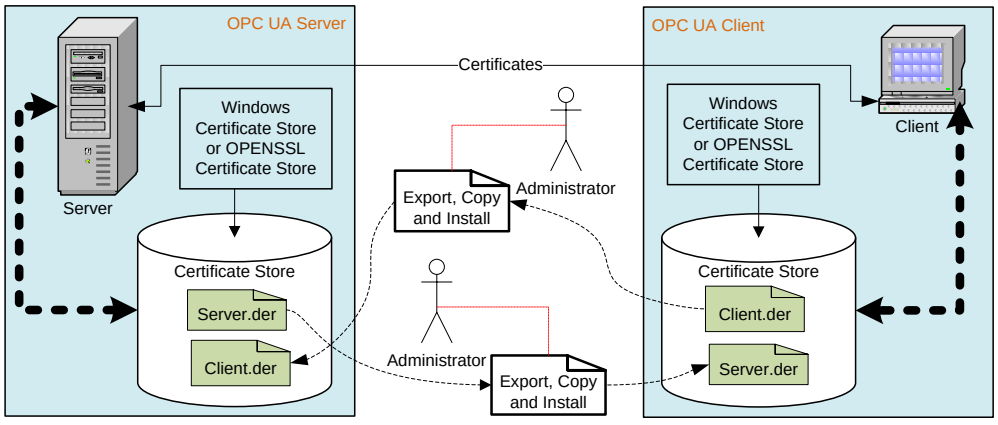


Figure 12 - Manual Certificate Handling

5.3.2 Case 2 – Windows Workgroup

Let's assume the system is a group of PCs running Windows Operating systems, with all PCs in the same workgroup. The PCs do NOT have access to a domain controller and the network is not connected to the internet. The PCs are provided by multiple vendors with unique configurations.

Tier 1 – would require no additional effort on the part of an end user. Most vendor provided clients and servers would automatically provide the self-signed Certificate that is required for this type of installation.

Tier 2, Tier 3, Tier 4 – All include the same type of effort for an administrator, only the list of certificates would change. It is recommended that a CA is defined in the system using the OPC Foundation tool. The administrator for each application as they are deployed would issue a request to the CA and obtain a certificate from the CA. The request would require interacting with the PC which has been designated as the CA. They would also install on each application PC the CA certificate as a trusted CA and would install any CRL lists. If a CA is not desired then as an alternative the Self-signed certificates can be copied via a manual certificate coping process. The OPC UA Configuration tool can assist with this operation by exporting and importing certificates and easily allowing them to be added to an application's trust list, but the complete list of trusted certificates must be copied to each machine.

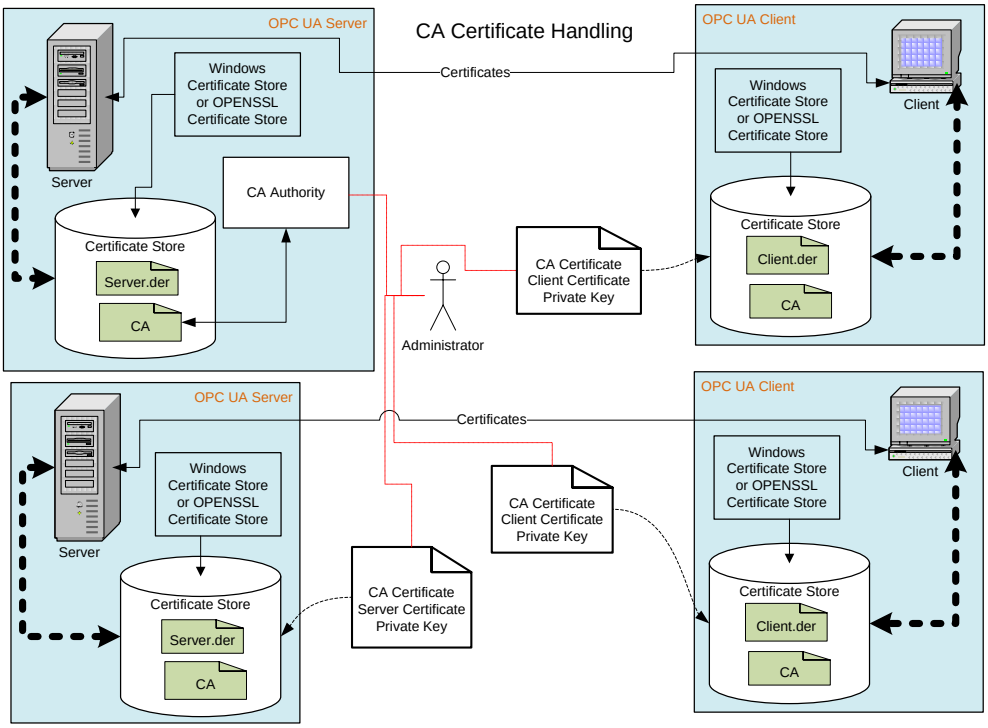


Figure 13 - CA Certificate Handling

5.3.3 Case 3 – mixed network

Let's assume that the environment is a mixed environment, one in which both clients and servers exist on both windows and non-windows platforms. In this case Tier 1 is again not an issue so let's concentrate on Tiers 2, 3 and 4.

All of the items described in the windows environment can still be used for the windows machines. The OPC foundation also provides a Certificate Generator that could be used in a non-windows environment. This Certificate Generator is based on OpenSSL and can be built and used on non-windows platforms. The Manual steps listed in Case 2 – Windows Workgroup also work on non-windows machines. The OpenSSL library may also be directly used on a non-windows machine. In general, using a CA in mixed environment is highly recommended since an Administrator would

otherwise be required to learn multiple tools. If a CA is used it can be based on a windows platform and the OPC Foundation provided tool can be used to manage it. The resulting Application instance certificates can be copied to each machine (windows or non-windows).

6 References

[UA Part 2] OPC UA Specification: Part 2 – Security 1.01

http://www.opcfoundation.org/UA/Part2/

[UA Part 4] OPC UA Specification: Part 4 – Services 1.01

http://www.opcfoundation.org/UA/Part4/

[UA Part 6] OPC UA Specification: Part 6 – Mappings 1.01

http://www.opcfoundation.org/UA/Part6/

7 Appendix – OpenSSL.CNF File

Below is a sample of the OpenSSL configuration file that is used for the commands list in section 4.6. The latest version of this file should be obtained from the OpenSSL site and review / updated as needed.

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 certificates 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 different ways 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