Kepware Technologies

Configuring Secured UA Communications Using ClientAce

June, 2013
Ref. 1.01

Table of Contents

1. Overview

This document intends to discuss OPC Unified Architecture (UA) and describe how users can create OPC UA secured communications using ClientAce and KEPServerEX.

2. OPC UA Secured Communications

OPC UA is an open standard created by the OPC Foundation with help from dozens of other member organizations. It intends to provide a platform-independent interoperability standard in order to move away from Microsoft COM; however, it is not a replacement for OPC Data Access (DA) technologies. For most industrial applications, UA will enhance existing OPC DA architectures. It will not be a system-wide replacement. UA complements OPC DA infrastructures in the following ways:

- It offers a secure method for client-to-server connectivity without depending on Microsoft DCOM, and has the ability to connect securely through firewalls and over VPN connections. For users connecting to remote computers within the corporate network (inside the firewall) on a domain, an OPC DA and DCOM connection may be satisfactory.
- It provides an additional way to share factory floor data with business systems (shop-floor to top-floor). OPC UA can aggregate data from multiple OPC DA sources into non-industrial systems.

In the majority of user applications, the most relevant components of OPC UA are as follows:

- Secure connections through trusted certificates for client and server endpoints.
- A robust item subscription model that provides efficient data updates between clients and servers.
- An enhanced method of discovering available information from participating UA servers.

ClientAce version 4.0 allows users to utilize a combination of new and existing APIs to incorporate both secured and unsecured OPC UA communication and functionality. This expanded functionality comes with additional APIs for PKICertificate creation, encapsulation, and management within the Windows Certificate Store.

2.1 OPC Unified Architecture Security Model

In OPC UA secured communications, each installation of an application or client is known as an Application Instance and must supply a unique certificate to an OPC UA Server for authentication. This authentication process requires the OPC Server to determine the validity of the certificate and either accept or deny its request to open a secured connection. Configuration on both the server and client side is necessary in order to achieve this authentication process.

2.1.1 OPC UA Security Options

The OPC UA security model incorporates three different security policies: None, Basic128Rsa15, and Basic256. Each policy gives users the option to define the message security mode that will be used during communications.

Descriptions of the policies are as follows:

- **None:** This policy does not use security. As such, there is no initial configuration necessary.
- **Basic128Rsa15:** This policy uses 128 bit cryptography for the Key Wrap Algorithm.
- **Basic256:** This policy uses 256 bit cryptography for the Key Wrap Algorithm. This setting is recommended for sensitive applications, and may soon be considered a minimum requirement.

Asymmetric encryption algorithms are used during secure OPC UA communications. This is accomplished by using both private and public keys, which are properties of both the client and server certificates. These keys are necessary for the encryption and decryption of information passed through a secured UA connection.

The Message Security Mode specifies the security measures that will be applied to messages. The values are enumerated. For more information, refer to the table below.

Mode	Value	Description
None	0	No security is applied. In some cases, the Security Policy will be ignored.
Sign	1	All messages between the client and server are signed, but not encrypted.
Sign and Encrypt	2	All messages between the client and server are signed and encrypted.

2.2 Using ClientAce APIs for Certificate Handling and Creation

ClientAce version 4.0 targets the 4.0 .NET Framework and allows users to easily create, store, retrieve, and use PKICertificate instances for secured UA connections. This increased functionality comes through the addition of the PkiCertificate Class to the ClientAce API, which provides useful encapsulation of a certificate instance in a .NET environment. Certificate storage is completed through the Windows Certificate Store. Programmers choose both the store and the location in which the certificates will be stored and retrieved. The PkiCertificate Class is located beneath the Kepware.ClientAce.OpcCmn Namespace.

3. Creating a UA Secured Connection between KEPServerEX and ClientAce

3.1.1 Disabling the Firewall for TCP Ports

Before starting, users should check to ensure that the firewall for both the client and server computers allow communications through the proper TCP ports. The default setting for KEPServerEX is 49320.

When using a local OPC UA Discovery Server, communications must be enabled through port 4840. For initial testing, it is also helpful to determine whether proper functionality exists between the client application and the OPC UA Server (using a non-encrypted connection channel). This can be traced using a packet-capturing program like Wireshark.

3.1.2 Creating a PKICertificate Instance

Users can easily create a PKICertificate instance in code by using the constructor demonstrated below in Visual Basic .NET.

```
clientCertificate = New Kepware.ClientAce.OpcCmn.PkiCertificate("Application URI", _
“127.0.0.1”, _ ‘(IP Address where this application will run)
"HostName", _
Integer.MaxValue, _ ‘(Time in seconds this certificate will stay valid)
"Name_Of_My_Certificate", _
"Kepware Technologies", _
"Development", _
"Portland", _
"Maine", _
"United States", _
 “2048”) ‘ (The length in bits of the key used for decryption)
```

This new PKICertificate instance contains all the necessary information required to utilize, store, and retrieve the certificate for future use.

3.1.3 Important Certificate Attributes

A newly created certificate instance has three important member variables: the Certificate Thumbprint, the Certificate Application URI, and the Private Key. Descriptions of the variables are as follows:

- **Certificate Thumbprint:** This is used to identify and retrieve the certificate from the Windows Certificate Store.
- **Certificate Application URI:** This is used as a substitute for the Certificate Thumbprint in order to identify and retrieve the certificate from the Windows Certificate Store.
- **Private Key:** This is used to decrypt communications between the server and client application.

Note: If a secure UA connection is required, the Private Key must be included within the connection object even if encryption is not used.

3.1.4 Sending the Certificate to the Windows Certificate Store

Once the certificate has been created, users must send it to the Windows Certificate Store for later retrieval. The ClientAce API supports two methods for doing so: “<*TheClientCertificate*>.toWindowsStore” and “<*TheClientCertificate*>.toWindowsStoreWithPrivateKey”. As the methods’ names suggest, users have the option of keeping the Private Key with the certificate or managing it separately.

Note: It is recommended that password protection be used if the Private Key is included with the certificate. Because this is not a feature of the ClientAce API, it needs to be handled using an external certificate management program like the Certificate Snap-In within the Microsoft Management Console (MMC).

The two arguments used for both the toWindowsStore Method and the toWindowsStoreWithPrivateKey Method are the Windows Store Location and the Windows Store Name. Descriptions are as follows:

- **Windows Store Location:** This is a constant defined within the API that is used to specify predetermined locations. The valid store locations are defined in the table below.

CurrentService	262144
CurrentUser	65536
LocalMachine	131072
Services	327680
Users	393216

Note: The most common Windows Store Locations are CurrentUser and LocalMachine. All locations require Administrator permissions or additional configuration for use (except for CurrentUser).

- **Windows Store Name:** This is a user-defined folder within a store location. Most OPC UA applications use the store name "UA Applications". Store locations that do not already contain a Store Name specified by the user will have one created automatically.

3.1.5 Retrieving the Certificate from the Windows Store

If the Windows Store Location, Windows Store Name, and Application URI (or the Certificate Thumbprint) are known, a client application can retrieve a certificate with or without the Private Key using the following methods:

- Kepware.ClientAce.OpcCmn.PkiCertificate.fromWindowsStoreWithPrivateKey Method
- Kepware.ClientAce.OpcCmn.PkiCertificate.fromWindowsStore Method

A PKICertificate instance will be returned once the certificate is found at the specified location.

3.1.6 Passing Certificates To and From the ConnectInfo Object

Once users have a valid PKICertificate instance and the Private Key, they can begin populating the ConnectInfo Object, which is necessary for establishing a connection when using the ClientAce APIs. The ConnectInfo Object requires a DER-encoded byte array for both the server and client certificates.

To do this conversion, simply use the <*PKICertificate*>.toDER Method. Users that already have a DER-encoded byte array version of the certificate can create a PKICertificate instance using the fromDER (static) Method.

3.1.7 Retrieving a Server's Certificate

Users can retrieve the server certificate from the server through the EnumComServer Method, getEndpoints Method, or getCertificateForEndpoint Method.

Note: The EnumComServer Method can only be used to retrieve information from an OPC UA Server when a Local Discovery Server is used. Otherwise, the getEndpoints or getCertificateForEndpoint Methods can be used.

After the server's UA Certificate has been retrieved successfully, users must store it within the Windows Store (preferably within the "UA Applications" Store Name). For more information, refer to the instructions below.

1. Use the fromDER Method to create a PKICertificate instance of the server certificate retrieved from the methods described above.
2. Then, use the <*serverCertificate*>.ToWindowsStore Method to send it to the Windows Store.

   Note: Alternatively, users can export KEPServerEX's UA Server certificate from the OPC UA Configuration Manager and then manually import it to the appropriate Windows Store Location.

Using the getEndpoints Method provides users the ability to retrieve all UA endpoints from a UA server along with their respective Security Policy URIs and Message Security Modes. All three of these methods return a DER-encrypted certificate that can then be directly passed to the ConnectInfo Object.

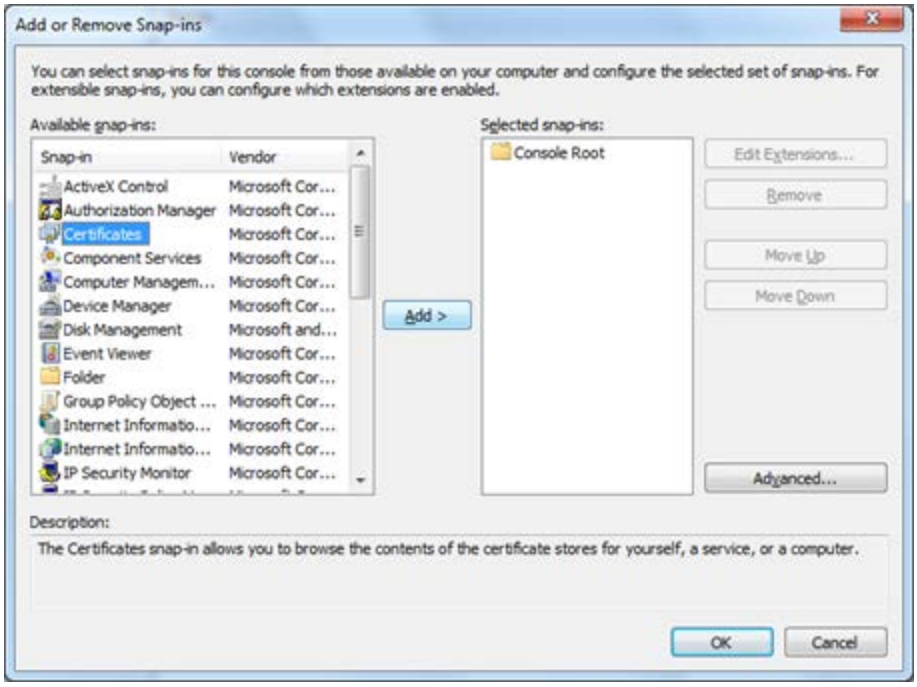
3.2 Establishing a Secure UA Connection

Once users have the required certificate and connection information, they must make sure that the remote UA server (in this case, KEPServerEX) is configured to accept the client application's request for establishing a secured connection.

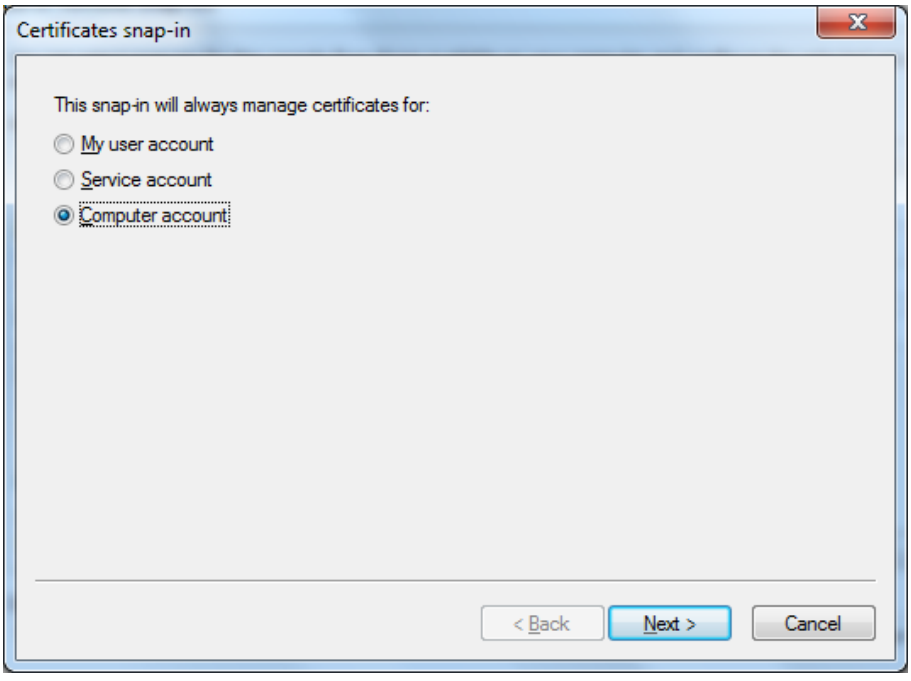
3.2.1 Trusting the Client Application

To complete the trading of certificates, users must supply KEPServerEX with the certificate from the client application and then trust it. This functionality is built into KEPServerEX's OPC UA Configuration Manager. For more information, refer to the instructions below.

1. To start, export the Client Application Certificate from the Windows Store and give it to KEPServerEX to trust. To do so, use the **Certificate Snap-In** from the **Microsoft Management Console (MMC)**.
2. Navigate to the *C:\Windows\System32* directory and find **mmc.exe**.
3. Then, open **mmc.exe** and click **File** | **Add or Remove Snap-ins**.
4. Select **Certificates**, and then click **Add**.

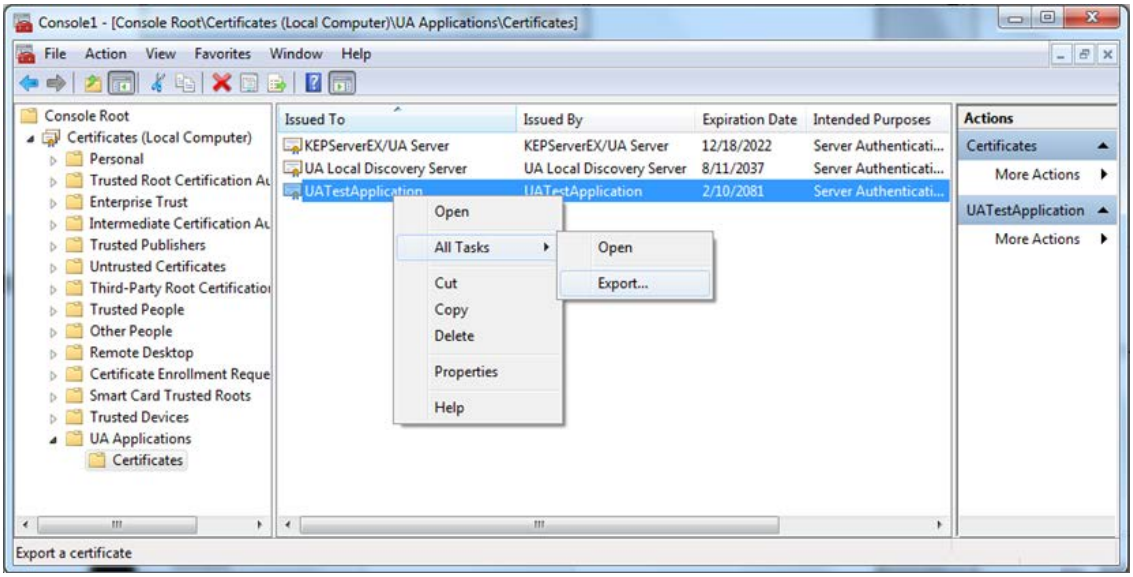


5. In **Certificates snap-in**, select the type of certificate that will be managed. Options include **My user account**, **Service account**, or **Computer account**. The selection depends on whether the client application is running for a specific user, as a service, or is available to the whole computer.

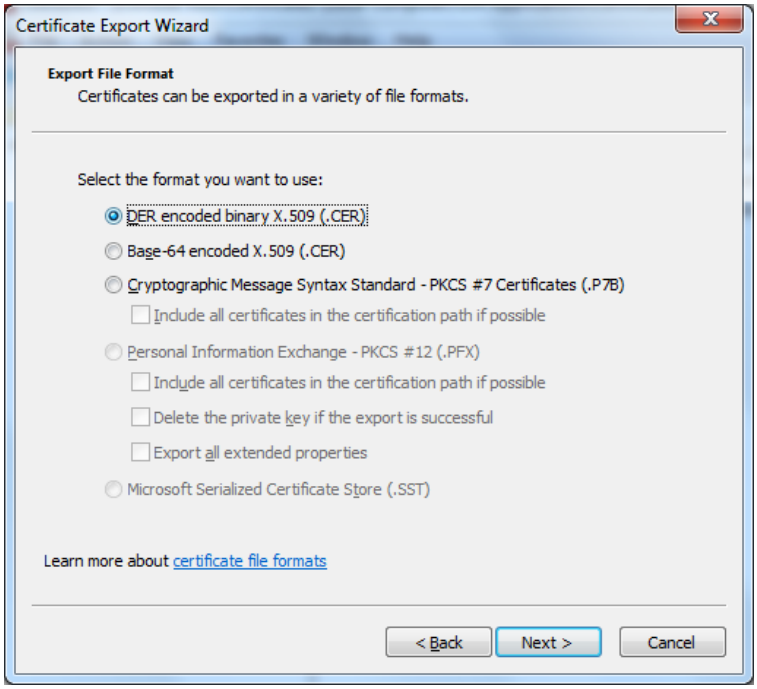


Note: If the wrong type of snap-in is chosen, another may need to be added (or multiple may need to be added simultaneously).

6. Once the snap-in has been added, export the client application's certificate. To do so, find and select the Store Location in which the client certificate resides.
7. Then, right-click on the certificate and select **All Tasks** | **Export**.



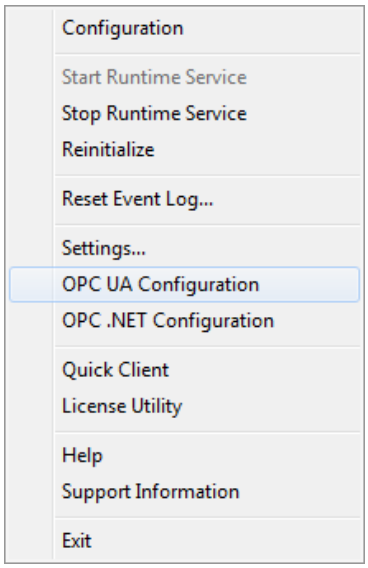
8. In the **Certificate Export Wizard**, select a file format and destination for the exported certificate.



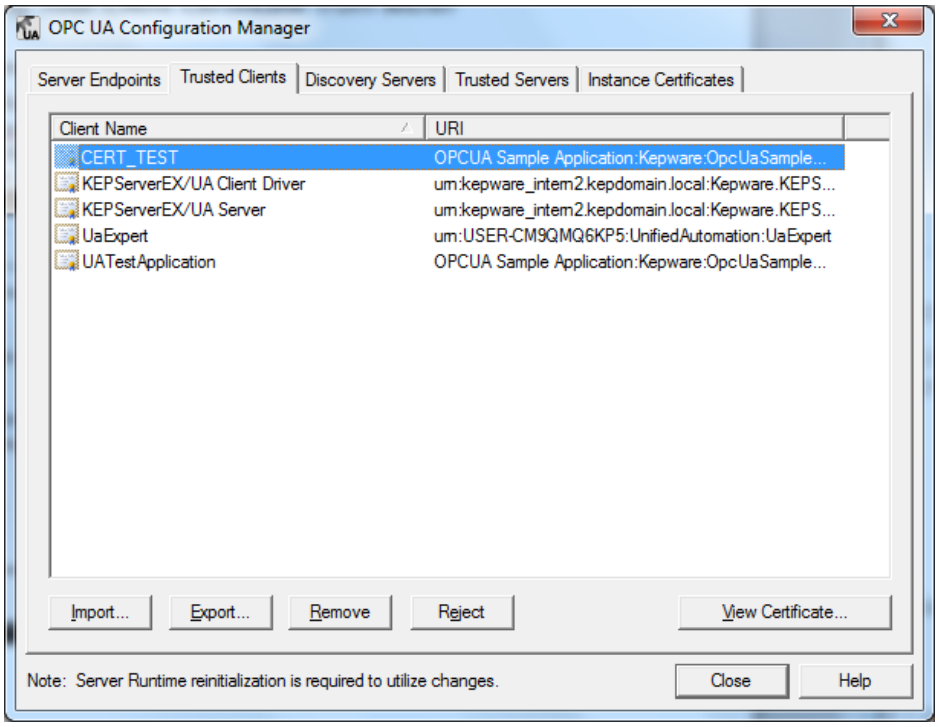
Note: The Private Key is not needed for KEPServerEX to trust and extend a secure UA Communications Channel. As such, the file format .CER is acceptable for use as a DER-encoded Binary.

9. Once the certificate is exported, users can configure KEPServerEX to trust it. To start, copy the certificate to the server machine.

10. Next, right-click on the **Administration** icon located in the **System Tray** and select **OPC UA Configuration**.



11. Open the **Trusted Clients** tab, and then click **Import**.



12. Browse to and locate the client certificate. Then, click **Open**.

Note: Users will be notified as to whether the certificate import succeeded or failed. If no additional configuration is necessary, users must restart the KEPServerEX Runtime before the changes made within the OPC UA Configuration Manager can take effect.

3.2.2 Configuring the OPC UA Server Endpoints

At this point, the secured UA Server Endpoints can now be configured. For more information, refer to the instructions below.

1. In the **OPC UA Configuration Manager**, open the **Server Endpoints** tab.
2. To add a new server endpoint, select **Add**. To edit an existing server endpoint, click **Edit**.
3. In **Port Number**, select the TCP Port Number to use for the endpoint.
4. In **Security Policies**, select the appropriate security policies in addition to the Message Security Modes that are available for this particular endpoint.
5. Once finished, select **OK**. Then, click **Close**.
6. Restart the KEPServerEX Runtime in order for the changes to take effect.

3.2.3 Creating a Secured OPC UA Connection

With the client and server certificates on both the client and server machines (and the ConnectInfo Object containing both the client and server certificates with the Private Key for the client), users are now ready to pass this method to a DaServerMgt Object and use the Connect Method to establish a connection.

If a certificate problem or mismatch occurs, an exception will be returned from the DaServerMgt.Connect Method that states “An argument to the function was invalid”. Users can troubleshoot a Secured Connect Server Rejection by viewing KEPServerEX’s Event Log. For information on enabling the Event Log, refer to the instructions below.

1. In the KEPServerEX Configuration, click **File** | **Project Properties**.
2. Next, select the **OPC UA** tab and locate the **Server Interface** section.
3. In **Log Diagnostics**, select **Yes**. This feature may help users identify the reason that a secured connection failed.
4. Once connected, all methods of the DaServerMgt Object will function as usual.

4. Summary

At this point, users should have a better understanding of OPC UA and of how to configure UA secured communications using ClientAce and KEPServerEX.