A Ready To Use Software Licensing Solution in C#

TonyTonyQ

30 Nov 2016 MIT

Rate me: 4.93/5 (131 votes)

Here we will talk about one kind of solution for software licensing using digital signature, which is implemented in C#.

Download QLicense.zip - 233.6 KB

Introduction

[Update Highlights]

2016.11 Thanks for the comments to point out the missing part for validating UID in the demo. The bug was fixed, you may get latest codes from CodeProject or GitHub. No changes to the core library, keep core lib's version as 1.1, and Demo App was updated to version 1.2.

2016.9 Thanks for Member 12119377's comments on public key & private key files' protection, so I've modified the library and let the public key & private key files as embedded resource of the assembly to ensure they are unable to be replaced easily. Also update library version to 1.1.

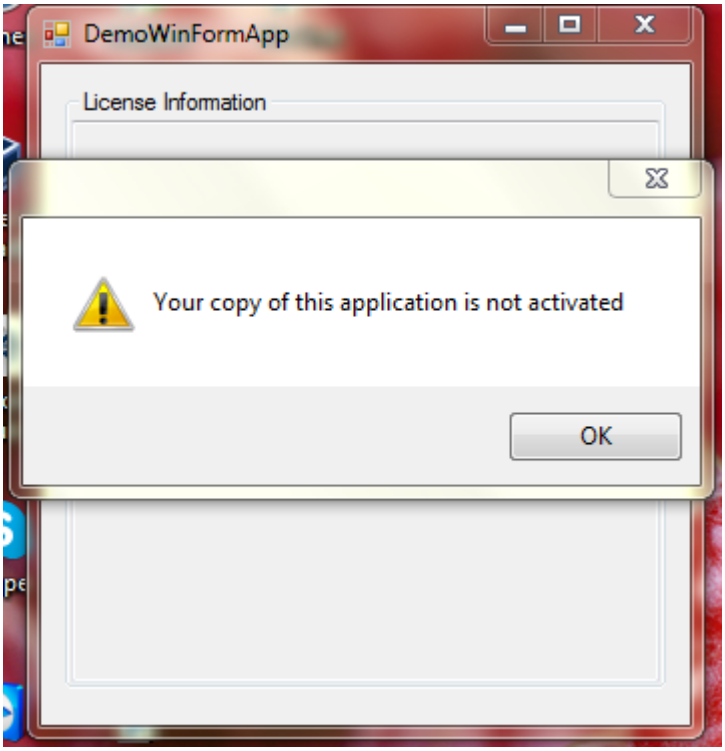
First, I'd like to show you a live demo for this solution.

A Simple Example

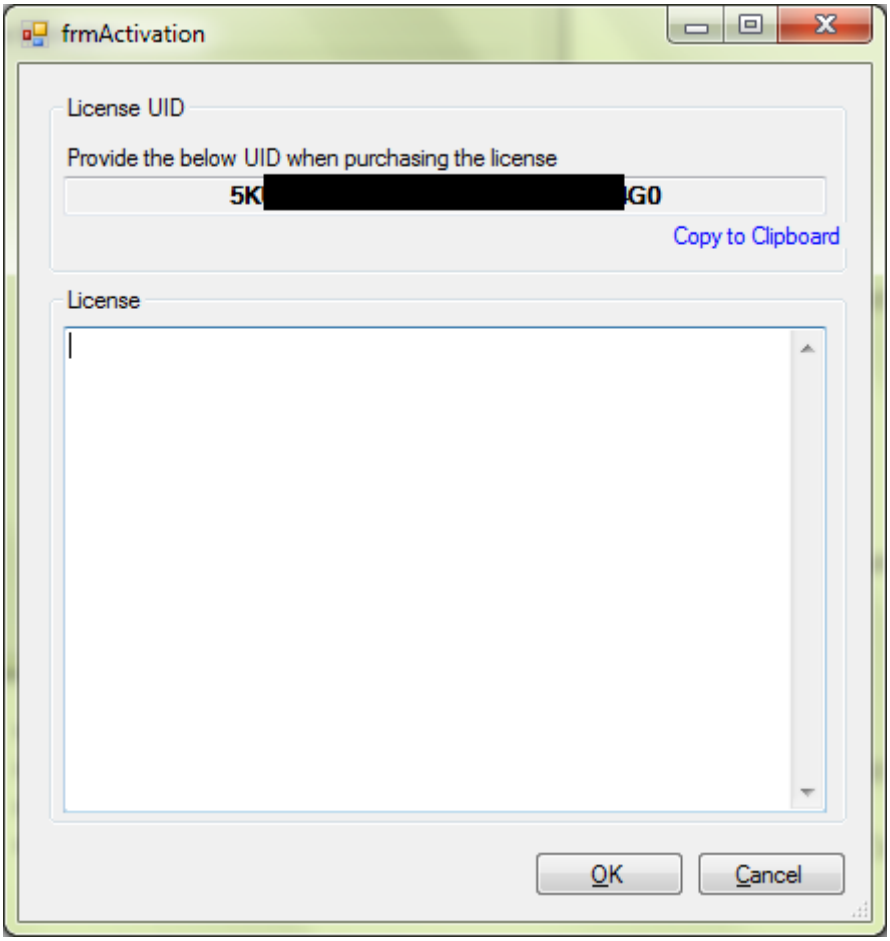
I had a application having feature 1, 2 and 3, and I want user to pay for each feature they like instead of paying for the whole package. So I designed a license solution to make this possible.

Current Solution

1) My application launches and finds itself not activated

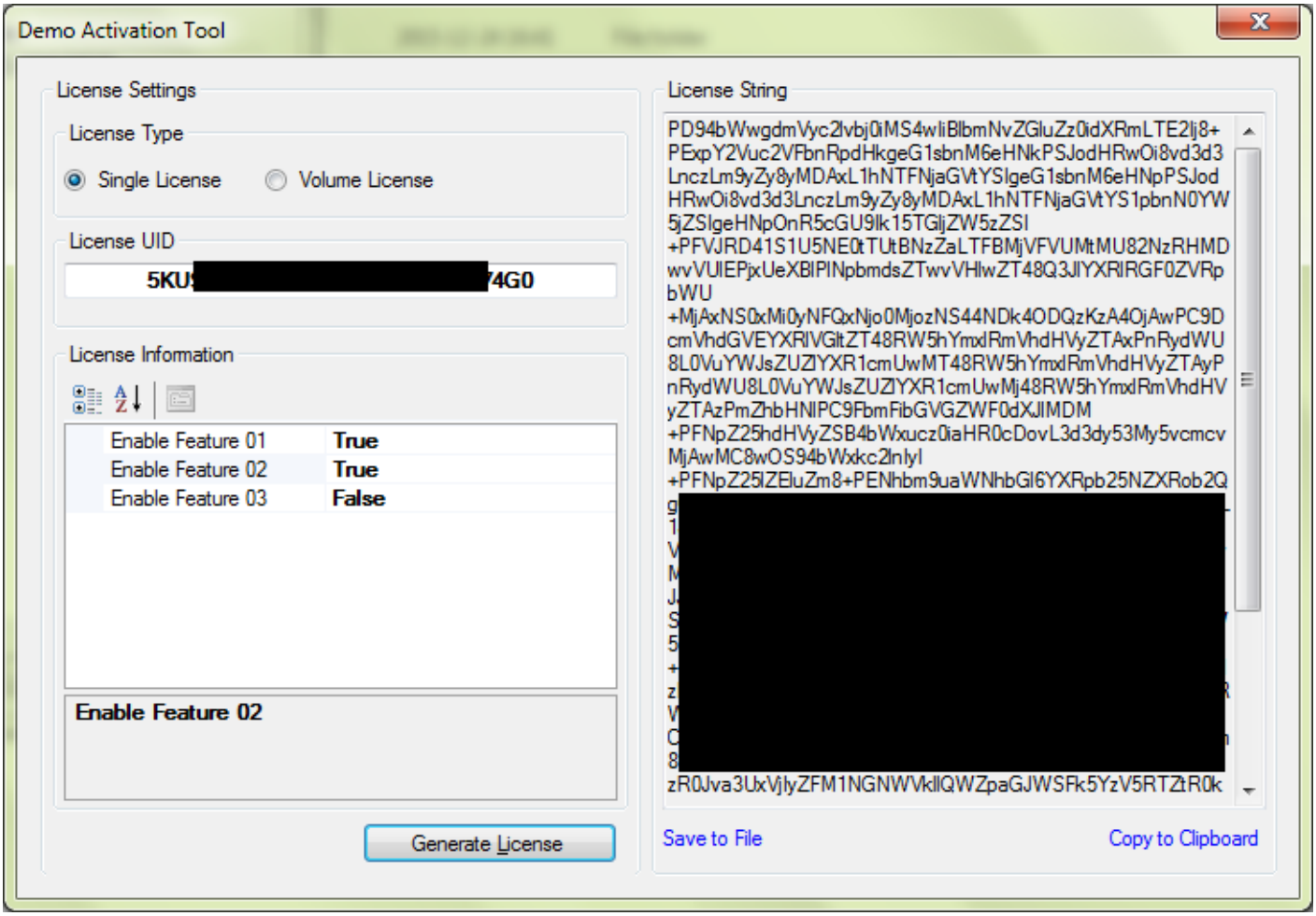


2) It displays the activiation form and show the current device UID for purchasing a license



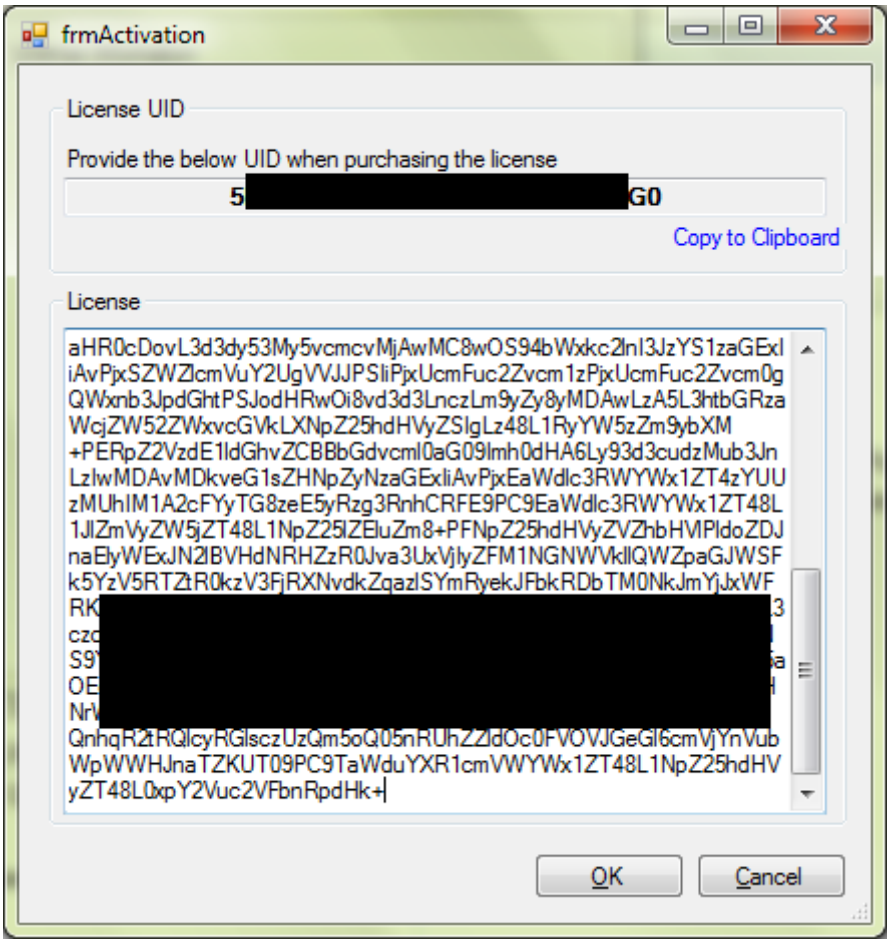
3) User shall give this device UID to our vendor to purchase the license, via either mail or phone.

4) At our vendor side, there is a license issuer program to issue the license based on the device UID and those features the user bought.



5) After license is generated, vendor shall paste the license string into a text file or mail to our user for activation.

6) User use paste the license string into the activiation form to activate the application.



Well, if this make you interested, please read on.

The highlights of this solution is:

- digital signature technology is used to protect the license file
- XML based license file allows it to contain rich information needed by the application
- support both single license mode and volume license mode.
- for single license mode, PC's unique key is generated based on PC's hardware. And BASE36 algorithm is used to generate the unique key for easier usage.

As I learnt, the key concept of such a license solution has been introduced years ago, even there are similar articles on CodeProject as well. However, it seems to be no ready-to-use solution or library out there. So it is a chance for me to do it!

Background

I was working on a project that is a system having many key features, which are expected to be actived or deactived based on the license provided. Boses want the feature on/off mechanism of the licensing system to be highly flexible, so the idea of pre-defined different editions is not suitable for this purpose.

The benefits is end user can just pay for those feature he/she want. E.g., user A can only pay for feature A, B & C; while user B only want feature A & C, and user C may only pay for feature B.

A good idea is to put those active features into a XML list and use this list as the license file. However, the next question is how to protect this license file since it is juat a plain text XML. Finally the digital signature solution answers the question.

Major Ideas

Main Workflow

The below chart describe the main workflow of this licensing solution:

Notes:

1. End User Application generates the Unique ID for the current PC.
2. Unique ID will be passed to License Issuer. This shall be a offline step which may include additional actions such as purchase and payment.
3. License Issuer issue the license file based on the Unique ID and license options. I.e., decide to enable which features based on the purchase and payment.
4. Send back the license file to end user. This is also a offline step, maybe by mails or USB drivers.
5. End User Application verify the license file and startup.

Unique ID for the device

For now, this solution is only used on PC, also including servers & laptops. It has not been used in mobile devices yet. I think the algorithm to get unique ID for mobile devices shall be different. Here we just talk about PC first.

The unique ID for a PC contains 4 parts:

- Application Name
- Processor ID
- Motherboard Serial Number
- Disk Volume Serial Number of Drive C

The 4 parts are concatenated as strings, then checksum is generated via MD5. And BASE36 encoding is used to format the checksum into a string like "XXXX-XXXX-XXXX-XXXX" for easier reading and transferring.

Hide Copy Code

```
/// <summary>
/// Combine CPU ID, Disk C Volume Serial Number and Motherboard Serial Number as device Id
/// </summary>
```

```
/// <returns></returns>
public static string GenerateUID(string appName)
{
    //Combine the IDs and get bytes
    string _id = string.Concat(appName, GetProcessorId(), GetMotherboardID(),
GetDiskVolumeSerialNumber());
    byte[] _byteIds = Encoding.UTF8.GetBytes(_id);

    //Use MD5 to get the fixed length checksum of the ID string
    MD5CryptoServiceProvider _md5 = new MD5CryptoServiceProvider();
    byte[] _checksum = _md5.ComputeHash(_byteIds);

    //Convert checksum into 4 ulong parts and use BASE36 to encode both
    string _part1Id = BASE36.Encode(BitConverter.ToUInt32(_checksum, 0));
    string _part2Id = BASE36.Encode(BitConverter.ToUInt32(_checksum, 4));
    string _part3Id = BASE36.Encode(BitConverter.ToUInt32(_checksum, 8));
    string _part4Id = BASE36.Encode(BitConverter.ToUInt32(_checksum, 12));

    //Concat these 4 part into one string
    return string.Format("{0}-{1}-{2}-{3}", _part1Id, _part2Id, _part3Id, _part4Id);
}
```

XML based license file

So, in order to let the license file contain more information, a XML based license file is necessary. In C#, we can easily use XML Serializer to serialize the object into XML and vice verca.

The basic license entity is defined as below: (*Some codes may contains Chinese, will do multiple language later*)

Hide Copy Code

```
public abstract class LicenseEntity
   {
       [Browsable(false)]
       [XmlIgnore]
       [ShowInLicenseInfo(false)]
       public string AppName { get; protected set; }

       [Browsable(false)]
       [XmlElement("UID")]
       [ShowInLicenseInfo(false)]
       public string UID { get; set; }

       [Browsable(false)]
       [XmlElement("Type")]
       [ShowInLicenseInfo(true, "Type", ShowInLicenseInfoAttribute.FormatType.EnumDescription)]
       public LicenseTypes Type { get; set; }

       [Browsable(false)]
       [XmlElement("CreateDateTime")]
       [ShowInLicenseInfo(true, "Creation Time",
ShowInLicenseInfoAttribute.FormatType.DateTime)]
       public DateTime CreateDateTime { get; set; }

       public abstract LicenseStatus DoExtraValidation(out string validationMsg);

   }
```

It contains 3 default properties:

- UID, which is the device's unique ID (Used to identify the specified device for single license. Not used for volume license)
- Type, indicate it is a single license or volume license
- CreateDateTime, indicate when is the license created

Since this is an abstract class, you can just extend it for additional properties by inheriting it.

Protect the license file

OK, now we have the license file and enough information we need. The problem is that a XML file for licensing is very weak because anyone can modify it. Thus we need to introduce digital signature solution.

Generate a certificate

To use digital signature, first of all, we need a pair of RSA keys. Private key for system owner to sign the XML file. And the public key for end user's application to verify the signed XML license file.

A easy way is to use "makecert" command to generate the certificates which we need to sign and verify the XML license file. Details will be covered in the later part of this article, please keep reading!

Sign the license file

C# provides native support for digital signature, so it is very easy to sign the XML license file as below: (These codes are copied from Microsoft samples)

Hide Shrink Copy Code

```
// Sign an XML file.
// This document cannot be verified unless the verifying
// code has the key with which it was signed.
private static void SignXML(XmlDocument xmlDoc, RSA Key)
{
    // Check arguments.
    if (xmlDoc == null)
        throw new ArgumentException("xmlDoc");
    if (Key == null)
        throw new ArgumentException("Key");

    // Create a SignedXml object.
    SignedXml signedXml = new SignedXml(xmlDoc);

    // Add the key to the SignedXml document.
    signedXml.SigningKey = Key;

    // Create a reference to be signed.
    Reference reference = new Reference();
    reference.Uri = "";

    // Add an enveloped transformation to the reference.
    XmlDsigEnvelopedSignatureTransform env = new XmlDsigEnvelopedSignatureTransform();
    reference.AddTransform(env);

    // Add the reference to the SignedXml object.
    signedXml.AddReference(reference);

    // Compute the signature.
    signedXml.ComputeSignature();

    // Get the XML representation of the signature and save
    // it to an XmlElement object.
    XmlElement xmlDigitalSignature = signedXml.GetXml();

    // Append the element to the XML document.
    xmlDoc.DocumentElement.AppendChild(xmlDoc.ImportNode(xmlDigitalSignature, true));

}
```

Now we are ready to publish the license for for end users. The last 'optimization' is to flat the XML so I encoded it with BASE64 and put all the output into a plain text file as the license.

Verify the license file

When the license file file is installed on the end user's PC, the end user's application shall populate the public key file and verify the license file using it.

Hide Shrink Copy Code

```
// Verify the signature of an XML file against an asymmetric
// algorithm and return the result.
private static Boolean VerifyXml(XmlDocument Doc, RSA Key)
{
    // Check arguments.
    if (Doc == null)
        throw new ArgumentException("Doc");
    if (Key == null)
        throw new ArgumentException("Key");

    // Create a new SignedXml object and pass it
    // the XML document class.
    SignedXml signedXml = new SignedXml(Doc);

    // Find the "Signature" node and create a new
    // XmlNodeList object.
    XmlNodeList nodeList = Doc.GetElementsByTagName("Signature");

    // Throw an exception if no signature was found.
    if (nodeList.Count <= 0)
    {
        throw new CryptographicException("Verification failed: No Signature was found in the
document.");
    }

    // This example only supports one signature for
    // the entire XML document.  Throw an exception
    // if more than one signature was found.
    if (nodeList.Count >= 2)
    {
        throw new CryptographicException("Verification failed: More that one signature was
found for the document.");
    }

    // Load the first <signature> node.
    signedXml.LoadXml((XmlElement)nodeList[0]);

    // Check the signature and return the result.
    return signedXml.CheckSignature(Key);
}
```

If the above verification is successful, the application starts to read the XML contents and can turn related features on/off accordingly.

About the Library

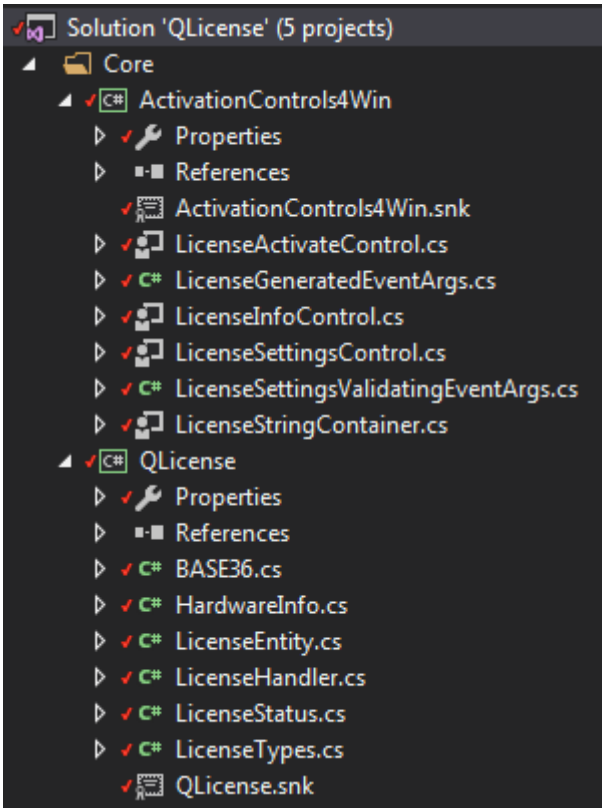
The whole .NET solution is built based on .NET Framework Version 4.0, in order to provide maximum compatibility even for retired Windows XP.

What I Have Provided

There are 5 projects inside this solution, which are split into 2 parts:

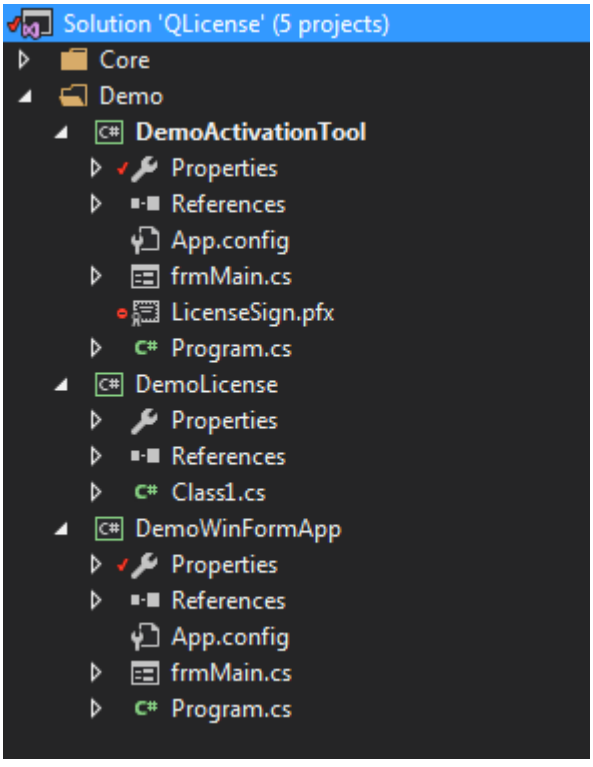
Core Libraries:

It contains 2 projects. One is QLicense which contains the core objects and logic. The other is ActivationControls4Win which contains the related Windows Form Controls for easy usage.



Demo:

This part contains a demo application which uses the above license solution and a demo activation tool which issues the license.



What You Need To Do

Step by Step is here for how to use this library. I will use those demo projects as a sample to describ how to create your own application that integrats this license solution.

1. Create your certificates for your application

It is recommended that you create a new set of certificates for each of your new application, this is for security consideration.

1) You can just use **makecert** command to do this like:

Hide Copy Code

```
makecert -pe -ss My -sr CurrentUser -$ commercial -n "CN=<YourCertName>" -sky Signature
```

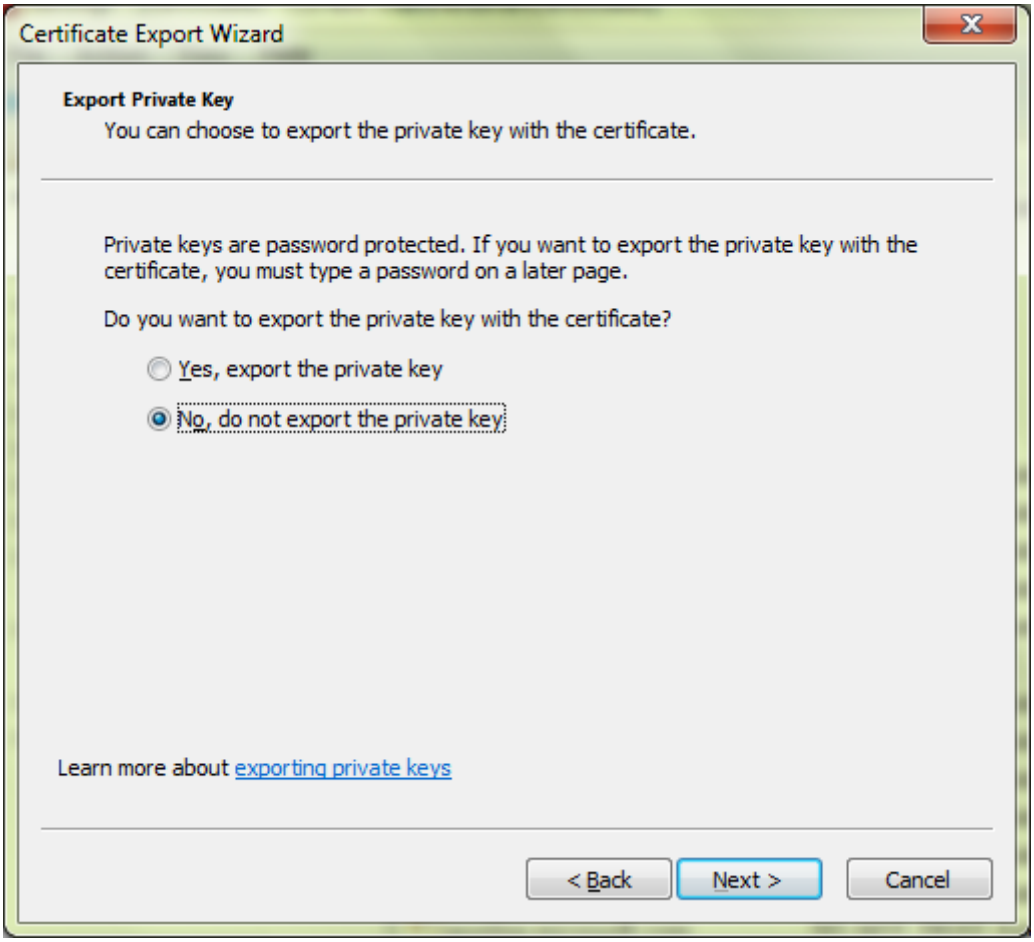
Replace "<YourCertName>" with your own certificate name.

2) After executing the above command, open your Certificate Management window by runninig "certmgr.msc".

3) Find the created certificate under "Personal" category with the name you specified in <YourAppName> above.

4) Right click on the certificate and select "All Tasks"->"Export"

5) On the Export dialogue, select "**Yes, export the private key**" and leave the other settings as default.



6) On the password diaglogue, input a password to protect the private key. You need to copy this password in your code when using this certificate. E.g., we use password "**demo**" here.

7) For the file name, we may use "**LicenseSign.pfx**" for this demo. Now we have the certificate with private key on hand.

8) Do step 4) to step 7) again to export the public key, just the difference is to choose "**No, do not export the private key**" and use file name as "**LicenseVerify.cer**" instead. Leave all the other options as default.

Well, we now have both the private key and public key generated for our solution.

2. Create your own license entity class - "DemoLicense"

1) You need to create a Class Library project which contains your own license entity, naming it "DemoLicense" for example.

2) Add Reference to **QLicense library**

3) Create a new class naming "MyLicense" and inherits "QLicense.LicenseEntity"

4) Add your own properties as you need

And the final codes shall look like:

Hide Shrink ▲ Copy Code

```
using QLicense;
using System.ComponentModel;
using System.Xml.Serialization;

namespace DemoLicense
{
    public class MyLicense : QLicense.LicenseEntity
    {
        [DisplayName("Enable Feature 01")]
        [Category("License Options")]
        [XmlElement("EnableFeature01")]
        [ShowInLicenseInfo(true, "Enable Feature 01",
ShowInLicenseInfoAttribute.FormatType.String)]
        public bool EnableFeature01 { get; set; }

        [DisplayName("Enable Feature 02")]
        [Category("License Options")]
        [XmlElement("EnableFeature02")]
        [ShowInLicenseInfo(true, "Enable Feature 02",
ShowInLicenseInfoAttribute.FormatType.String)]
        public bool EnableFeature02 { get; set; }

        [DisplayName("Enable Feature 03")]
        [Category("License Options")]
        [XmlElement("EnableFeature03")]
        [ShowInLicenseInfo(true, "Enable Feature 03",
ShowInLicenseInfoAttribute.FormatType.String)]
        public bool EnableFeature03 { get; set; }

        public override LicenseStatus DoExtraValidation(out string validationMsg)
        {
            //Here, there is no extra validation, just return license is valid
            validationMsg = string.Empty;
            return LicenseStatus.VALID;
        }
    }
}
```

Notes:

XmlElement attribute indicates the element name when the object is serilized into XML.

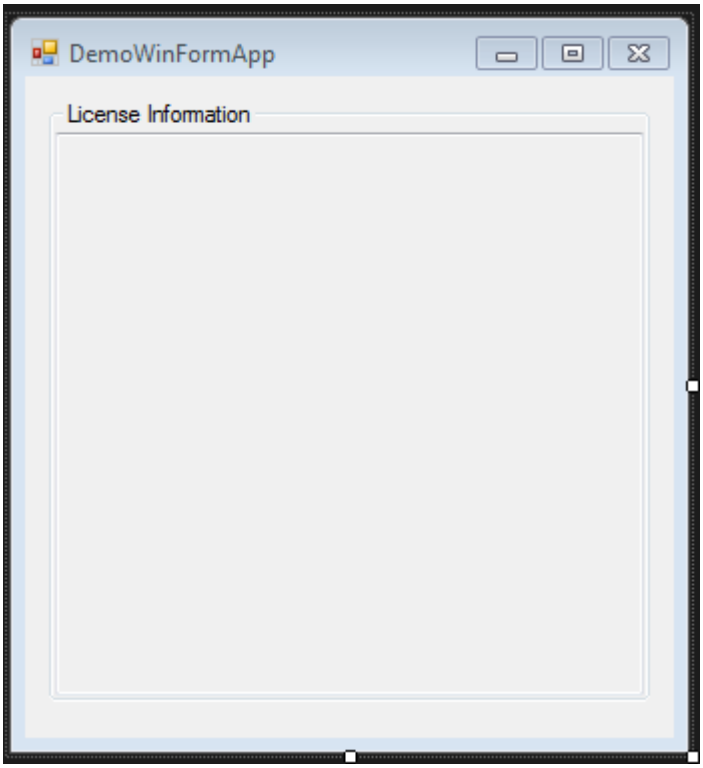
ShowInLicenseInfo attribute indicates whether this property shall be displayed in LicenseInfoControl which is a WinForm control contained in ActivationControls4Win project.

3. Integrate QLicense library with your application - "DemoWinFormApp"

1) Add Reference to **DemoLicense, QLicense** and **ActivationControls4Win**

2) Add **LicenseVerify.cer** generated in step 1 into the project, make it as **Embedded Resource** and **Do not copy to Output Directory** in the file property settings.

3) Put the **LicenseInfoControl** from **ActivationControls4Win** onto the form you'd like to show license details, such as About form. For the demo, I just put it on the main form.



4) The main logic for the main form is to validate the license and inform user to activate the application if needed. I put the logic in Form_Shown event in order to let main form shown in the backgound for better user experience. You may put these logic into a splash form or other place.

Hide Shrink Copy Code

```
        private void frmMain_Shown(object sender, EventArgs e)
        {
            //Initialize variables with default values
            MyLicense _lic = null;
            string _msg = string.Empty;
            LicenseStatus _status = LicenseStatus.UNDEFINED;

            //Read public key from assembly
            Assembly _assembly = Assembly.GetExecutingAssembly();
            using (MemoryStream _mem = new MemoryStream())
            {

_assembly.GetManifestResourceStream("DemoWinFormApp.LicenseVerify.cer").CopyTo(_mem);

                _certPubicKeyData = _mem.ToArray();
            }

            //Check if the XML license file exists
            if (File.Exists("license.lic"))
            {
                _lic = (MyLicense)LicenseHandler.ParseLicenseFromBASE64String(
                    typeof(MyLicense),
                    File.ReadAllText("license.lic"),
                    _certPubicKeyData,
                    out _status,
                    out _msg);
            }
            else
            {
                _status = LicenseStatus.INVALID;
                _msg = "Your copy of this application is not activated";
            }

            switch (_status)
            {
                case LicenseStatus.VALID:
```

```
                    //TODO: If license is valid, you can do extra checking here
                    //TODO: E.g., check license expiry date if you have added expiry date
property to your license entity
                    //TODO: Also, you can set feature switch here based on the different
properties you added to your license entity

                    //Here for demo, just show the license information and RETURN without
additional checking
                    licInfo.ShowLicenseInfo(_lic);

                    return;

                default:
                    //for the other status of license file, show the warning message
                    //and also popup the activation form for user to activate your application
                    MessageBox.Show(_msg, string.Empty, MessageBoxButtons.OK,
MessageBoxIcon.Warning);

                    using (frmActivation frm = new frmActivation())
                    {
                        frm.CertificatePublicKeyData = _certPubicKeyData;
                        frm.ShowDialog();

                        //Exit the application after activation to reload the license file
                        //Actually it is not nessessary, you may just call the API to reload
the license file
                        //Here just simplied the demo process

                        Application.Exit();
                    }
                    break;
            }
        }
```

4) Add a new form named **frmActivation** and put **LicenseActivateControl** from **ActivationControls4Win** onto it. This is the form user need to enter license content and activate your application.

5) Here is the main logic for **frmActivation.** It will display the calculated device UID once the form popup:

Hide Shrink Copy Code

```
    public partial class frmActivation : Form
    {
        public byte[] CertificatePublicKeyData { private get; set; }

        public frmActivation()
        {
            InitializeComponent();
        }

        private void btnCancel_Click(object sender, EventArgs e)
        {
            if (MessageBox.Show("Are you sure to cancel?", string.Empty,
MessageBoxButtons.YesNo, MessageBoxIcon.Warning, MessageBoxDefaultButton.Button2) ==
DialogResult.Yes)
            {
                this.Close();
            }
        }

        private void frmActivation_Load(object sender, EventArgs e)
        {
            //Assign the application information values to the license control
            licActCtrl.AppName = "DemoWinFormApp";
            licActCtrl.LicenseObjectType = typeof(DemoLicense.MyLicense);
            licActCtrl.CertificatePublicKeyData = this.CertificatePublicKeyData;
            //Display the device unique ID
            licActCtrl.ShowUID();
```

```
        }

        private void btnOK_Click(object sender, EventArgs e)
        {
            //Call license control to validate the license string
            if (licActCtrl.ValidateLicense())
            {
                //If license if valid, save the license string into a local file
                File.WriteAllText(Path.Combine(Application.StartupPath, "license.lic"),
licActCtrl.LicenseBASE64String);

                MessageBox.Show("License accepted, the application will be close. Please
restart it later", string.Empty, MessageBoxButtons.OK, MessageBoxIcon.Information);

                this.Close();
            }
        }
    }
```

4. Create your license issuer tool - "DemoActivationTool"

1) Create a new Windows Form Application project, naming "DemoActivationTool".

2) Add References to **DemoLicense, QLicense** and **ActivationControls4Win**

3) Add **LicenseSign.pfx** generated in step 1 into the project, make it as **Embedded Resource** and **Do not copy to Output Directory** in the file property settings.

4) Draw the control **LicenseSettingsControl** and **LicenseStringContainer** on the main form, it may look like:

5) Add the following codes for main form. Details are explains inline with the codes as comments below:

Hide Shrink ▲ Copy Code

```
    public partial class frmMain : Form
    {
        private byte[] _certPubicKeyData;
        private SecureString _certPwd = new SecureString();

        public frmMain()
        {
            InitializeComponent();

            _certPwd.AppendChar('d');
            _certPwd.AppendChar('e');
            _certPwd.AppendChar('m');
            _certPwd.AppendChar('o');
        }

        private void frmMain_Load(object sender, EventArgs e)
        {
            //Read public key from assembly
            Assembly _assembly = Assembly.GetExecutingAssembly();
            using (MemoryStream _mem = new MemoryStream())
            {

_assembly.GetManifestResourceStream("DemoActivationTool.LicenseSign.pfx").CopyTo(_mem);

                _certPubicKeyData = _mem.ToArray();
            }

            //Initialize the path for the certificate to sign the XML license file
            licSettings.CertificatePrivateKeyData = _certPubicKeyData;
            licSettings.CertificatePassword = _certPwd;

            //Initialize a new license object
            licSettings.License = new MyLicense();
        }
```

```
        private void licSettings_OnLicenseGenerated(object sender,
QLicense.Windows.Controls.LicenseGeneratedEventArgs e)
        {
            //Event raised when license string is generated. Just show it in the text box
            licString.LicenseString = e.LicenseBASE64String;
        }


        private void btnGenSvrMgmLic_Click(object sender, EventArgs e)
        {
            //Event raised when "Generate License" button is clicked.
            //Call the core library to generate the license
            licString.LicenseString = LicenseHandler.GenerateLicenseBASE64String(
                new MyLicense(),
                _certPubicKeyData,
                _certPwd);
        }

    }
```

That's all for this guide, the running result has been already shown at the beginning of this article.

Feel free to leave your comments and good ideas below.

History

Version 1.0: Released in May of 2015

Version 1.1: Released in Sep of 2016. For security consideration, embedded public key and private key file as embedded resource of the assembly instead of single local files.

About Source Codes

The source codes can be also found on:

- GitHub: https://github.com/soldierq/QLicense

- Codeplex: https://qlicense.codeplex.com/

License

Share

About the Author

TonyTonyQ
Architect
China
Follow this Member
No Biography provided

Comments and Discussions

Add a Comment or Question | Email Alerts | Search Comments

First Prev Next

Amazing piece of work and a really useful explanation above.
Member 14642295 15-Aug-20 21:42

Thank you,
HL7_Interface_Developer 21-Apr-20 23:18

Perfect One, but i need a wpf implemntation
Member 14001021 13-Apr-20 3:17

LicenseSign.pfx
Member 14702368 22-Jan-20 16:45

Invalid Algorithm Specified exception starting from .NET Framework 4.7.1
GIsidori 25-Nov-19 1:16

Re: Invalid Algorithm Specified exception starting from .NET Framework 4.7.1
z-boson 12-Sep-20 14:18

Adding Expiry Date to license entity?
Awais Haroon 16-Nov-19 10:12

Re: Adding Expiry Date to license entity?
icornato 3-Nov-20 9:29

Error in GetUIDInBytes(string UID)
Bastien Vandamme 15-Oct-19 6:52

Is that possible to use this in Web/Asp.core in IOS/Linux/Windows platform, Web Activation app running in centrelised ?
vijayakumar.p 16-Sep-19 15:16

It is possible to convert this license string in serial key format?
Member 12923819 30-Jun-19 12:34

LicenseSign.pfx Not Found
Member 12923819 13-Jun-19 15:05

Re: LicenseSign.pfx Not Found

David Fernando Cruz Benavidez 26-Jun-19 22:27

License Invalid
Member 14214320 5-Apr-19 15:35

Re: License Invalid
Member 10820940 3-Jan-20 11:34

How to include LicenseVerify.cer file into the project please
Member 14214320 5-Apr-19 14:29

Hi please some one tell me how i can use makecert command
Member 9959357 29-Jan-19 17:05

Re: Hi please some one tell me how i can use makecert command
suresh_fhills 6-Feb-19 12:56

Re: Hi please some one tell me how i can use makecert command
Bastien Vandamme 14-Oct-19 10:20

Password error
Stan3286 4-Jan-19 21:20

Re: Password error
TheThomasFlint 10-Jan-19 23:49

makecert obsolete now in necesary New-SelfSignedCertificate
Member 14080084 14-Dec-18 18:17

Re: makecert obsolete now in necesary New-SelfSignedCertificate
Bastien Vandamme 14-Oct-19 13:55

License is INVALID
Marcos Otilio 17-Oct-18 5:59

Re: License is INVALID
Member 14560514 14-May-20 12:40

Refresh 1 2 3 4 5 Next ▷

General News Suggestion Question Bug Answer Joke Praise Rant Admin

Use Ctrl+Left/Right to switch messages, Ctrl+Up/Down to switch threads, Ctrl+Shift+Left/Right to switch pages.

Permalink
Advertise
Privacy
Cookies
Terms of Use

Layout: fixed | fluid