Copy Protection and Licensing

Created: 2011-08-08

Last update: 2013-02-23

Table of Contents

Introduction to Copy Protection

When a user purchases your software you do not want unauthorized persons to use copies of your software when you distribute your software to the user. Limnor Studio has a built-in licensing system to protect the software you created using Limnor Studio.

Limnor Studio Copy-Protection system does not stop copying software because of following reasons:

- It is very difficult to prevent your software from being copied.
- The user may want to make copies for backup purpose.

Limnor Studio Copy-Protection system uses a licensing system to prevent unlicensed software from running. Your software can be copied to many computers. But each computer must get a license from you so that the software on that computer can run. One license is for one computer only.

Limnor Studio Copy-Protection system works in the following procedure:

1. Add a copy-protection component to your project
2. Develop your project and compile your software
3. Distribute your software to users
4. A user runs your software
5. Your software detects that it is unlicensed and asks the user for a serial number for the software. A license activation file is created.
6. The user sends the license activation file to you
7. You use Limnor Studio to process the activation file and generate a corresponding license file
8. You send the license file to the user
9. The user copies the license file to a specific folder.

Limnor Studio also provides an automated system so that you may use a web site to automatically process the license activation and deactivation, that is, steps 6, 7, 8, and 9 are automated. Your user clicks a button to do license activation or deactivation, the activation/deactivation message is sent to your web server and your web server connects to your license database and generates a license for your user and sends the license to your user's computer. Your software is ready to run on your user's computer immediately.

Once the license file is in the right folder, your software will run normally. The license file is only valid for the computer where the activation file is generated.

The licensing system allows specifying expiration time and software feature list. If an expiration time is given then the license will expire after the expiration time. Your software may validate that a given feature is activated for a license.

Each activation file and deactivation file can only be used once. Once a license is deactivated your user may re-activate his/her license on any supported computers.

Note that no copy-protection in the world is 100% bullet proof. It just makes it more difficult and costly for software piracy.

Add Copy Protection to Your Project

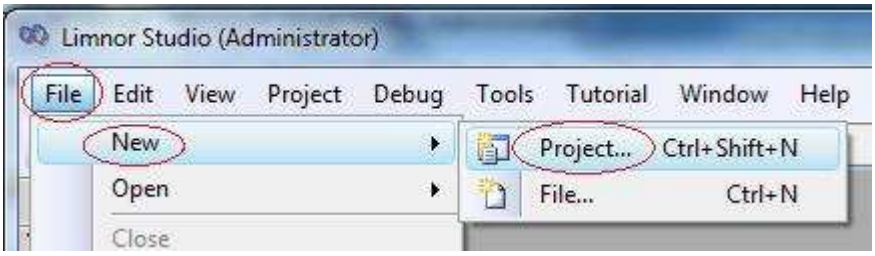
Use CopyProtector

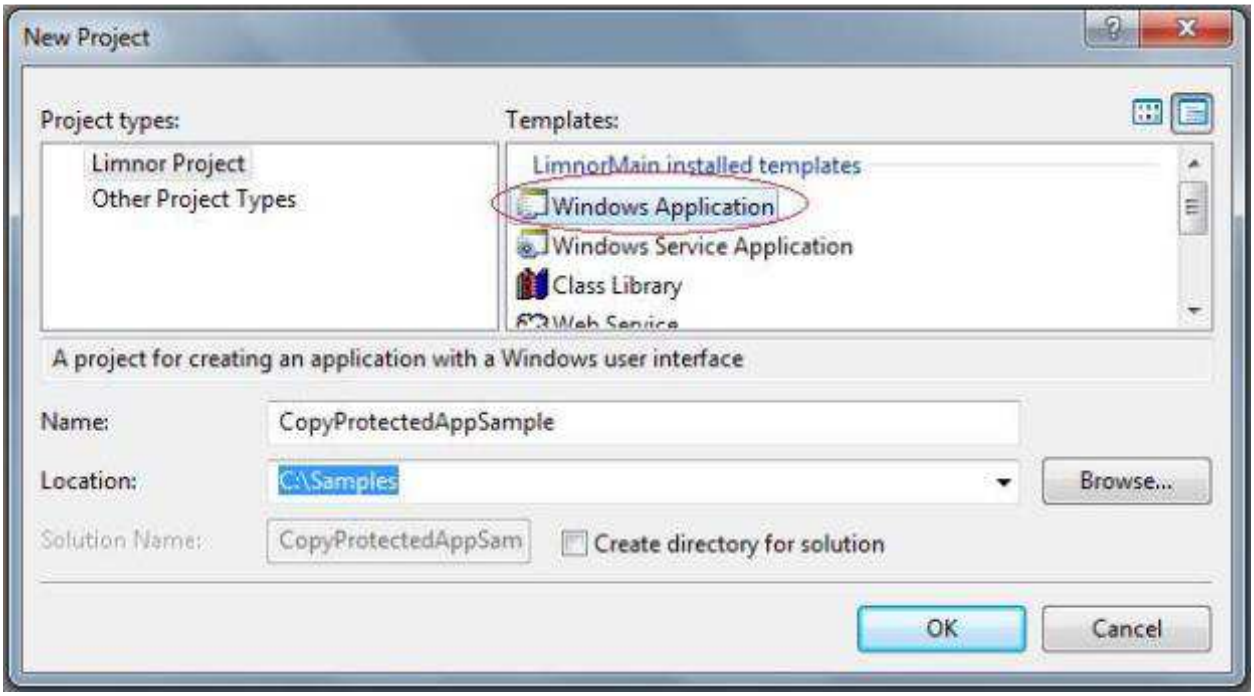
Limnor Studio provides a component named CopyProtector. It provides following methods for your software to protect itself.

- VerifyLicense – You may use this method to create an action. Assign this action to some events to execute it. When this action is executed if the software is not licensed then it will shutdown the software.
- CheckFeaturesAllowed – You may use this method to verify that a given list of features is activated by an installed license.
- Deactivate – You may use this method to allow your users to deactivate license so that a license may be activated on another computer.

Let's use a sample project to illustrate the use of CopyProtector. This sample project can be downloaded from http://www.limnor.com/studio/copyProtectedSample.zip

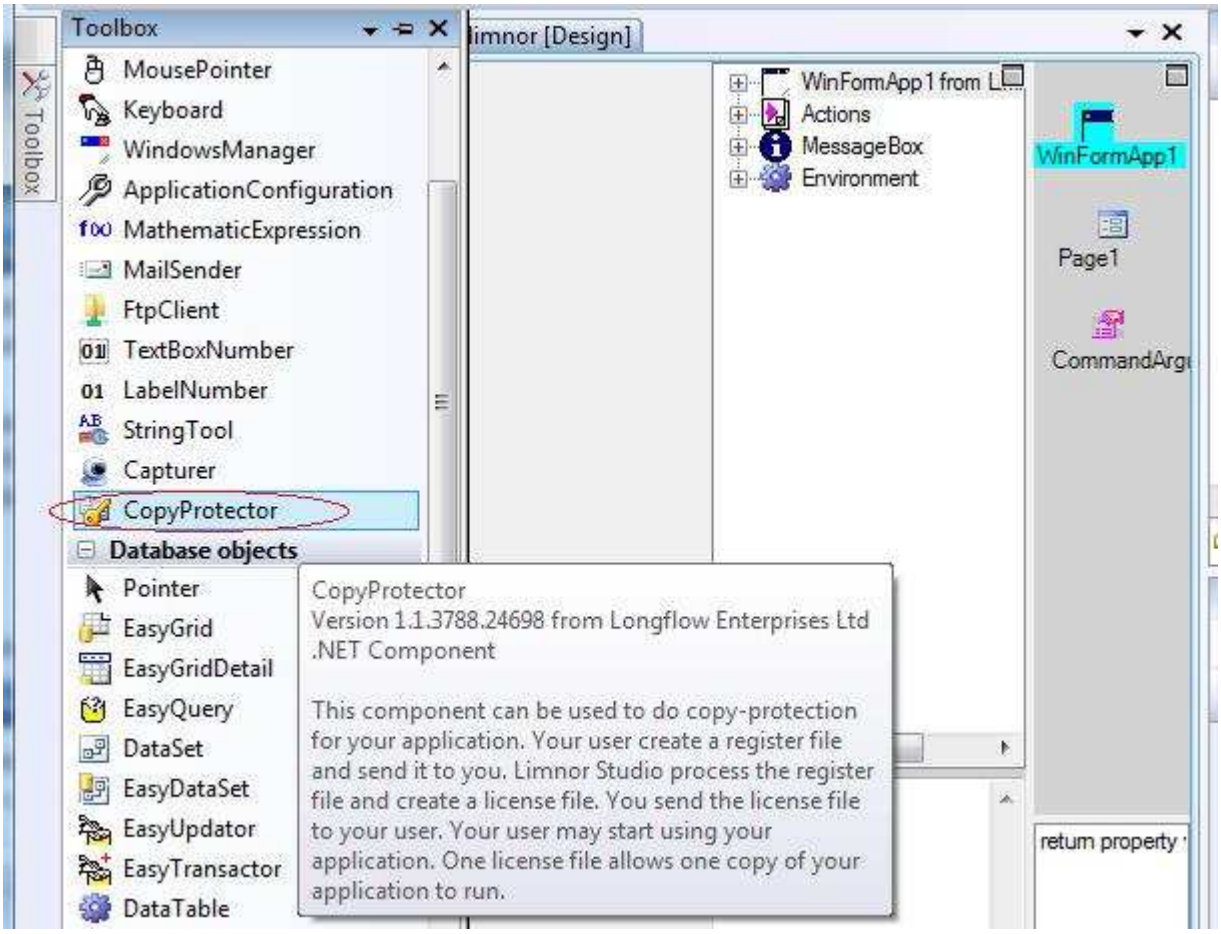
Create a Windows Form application.





Do not include underscore "_" in your application name. Underscores will be used in forming license file names.

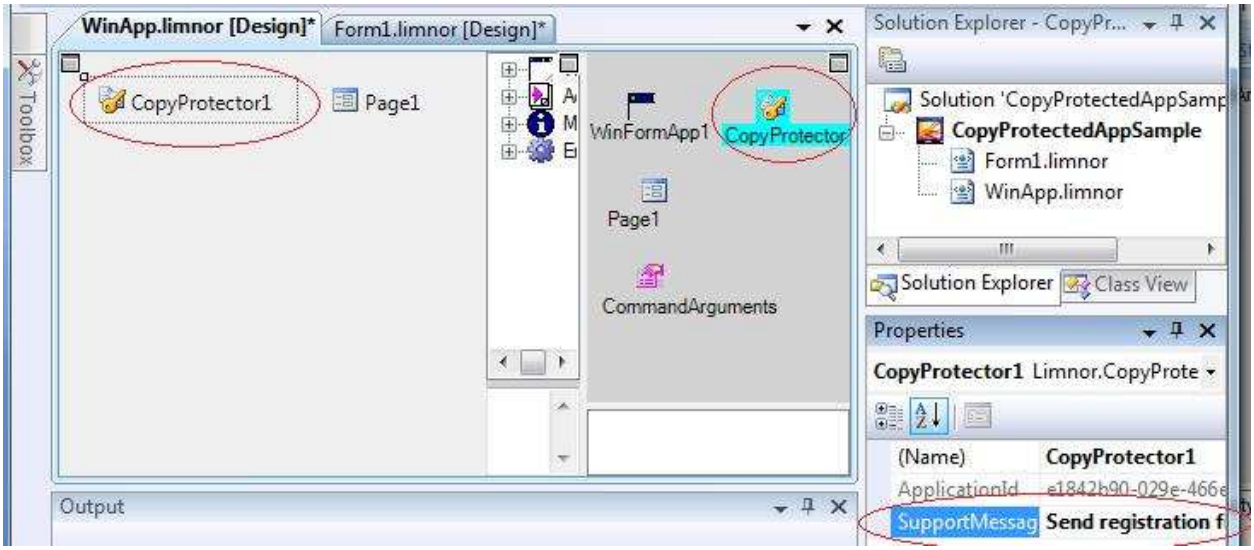
Drop CopyProtector to the WinFormApp1 class:



Set SupportMessage

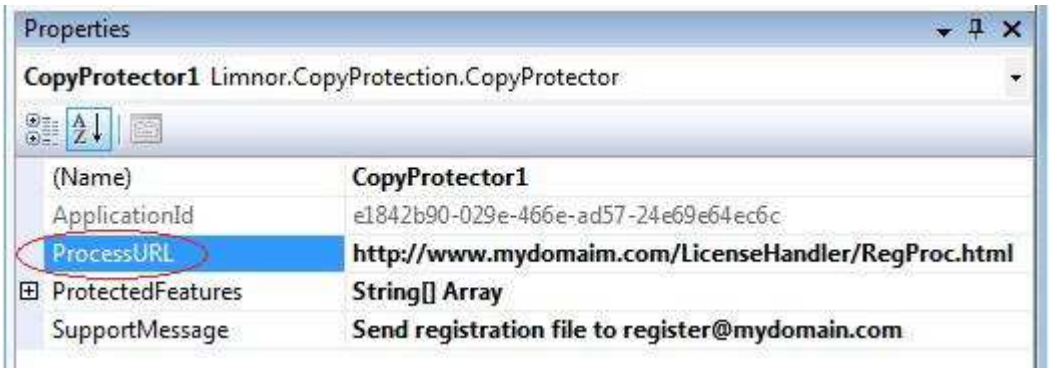
Property SupportMessage of the CopyProtector will be displayed to the user when it asks the user to create an activation/deactivation file. Usually you may use this property to notify the user where to send the activation/deactivation file. For example, we may set this property to

Send activation/deactivation file as an attachment to register@mydomain.com including software name and serial number, if automatic processing fails.



Set Default Activation Processing URL

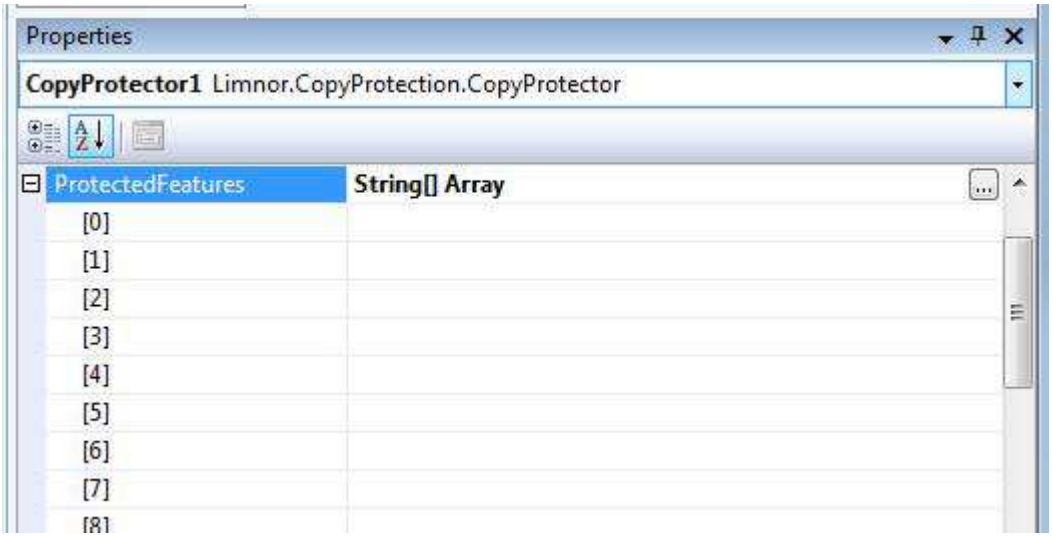
Limnor Studio provides a web page and its support files for automatically processing license activation and deactivation. The web page connects your licensing plug-in to know how you want to process each activation/deactivation request. This document will describe how to setup the web page on your web server. A sample licensing plug-in is also provided, which uses a database to manage all software licensing. You may set ProcessURL property to an URL pointing to the web page for processing activation/deactivation of licenses.



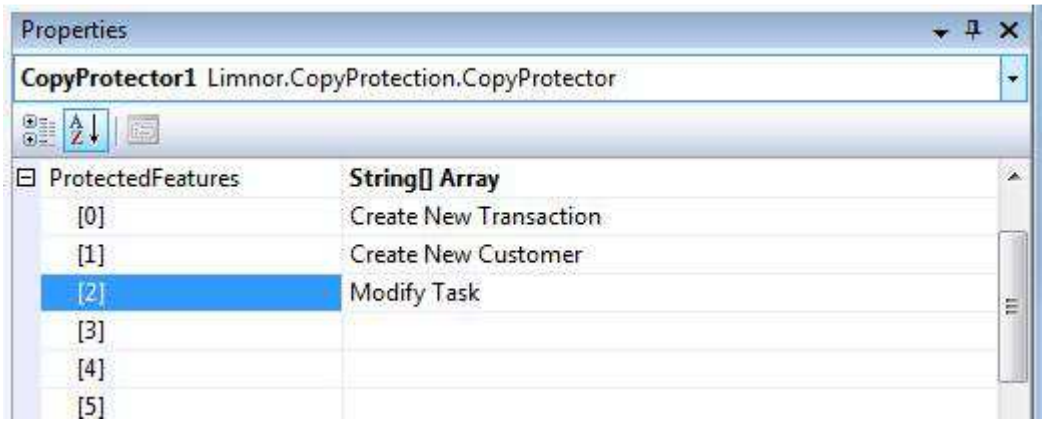
Set Feature List

You may protect up to 32 features in your software for one license. Each feature is identified by a number from 0 to 31.

To help you remember what features you want to protect you may give each feature a name and record feature names in property “ProtectedFeatures”:



For example, suppose you want to protect following features in your software: Create New Transaction; Create New Customer; and Modify Task. You may record them in ProtectedFeatures:

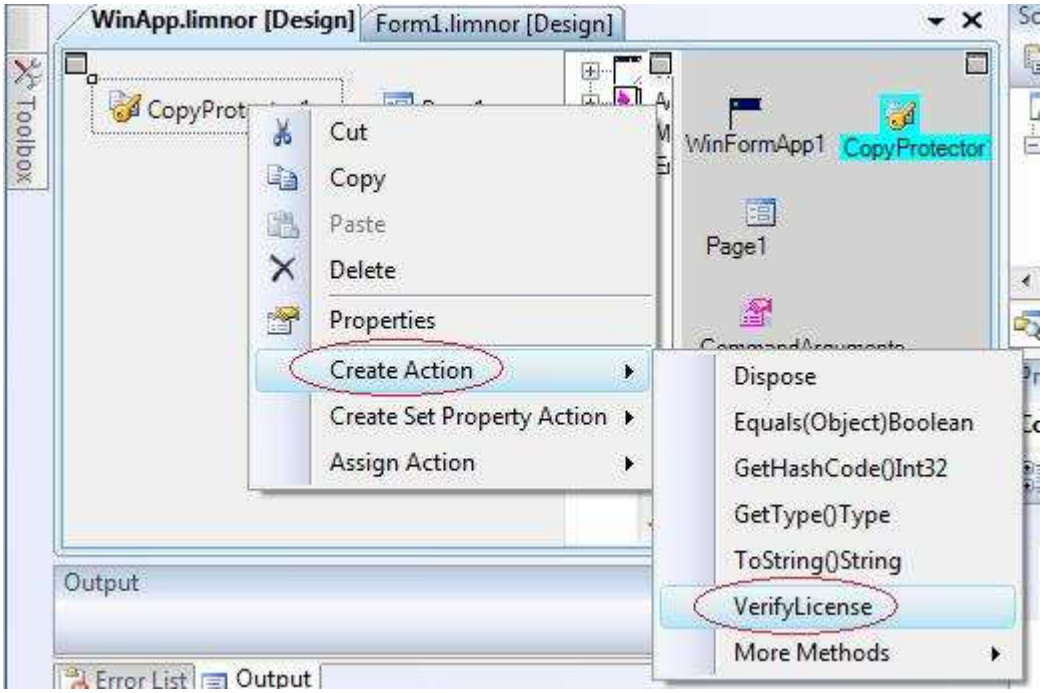


When you create a CheckFeaturesAllowed action to verify protected features it allows you to use a Boolean array to specify which features to verify. For example, suppose you want to verify that both "Create New Customer" and "Modify Task" are activated then you need to use a Boolean array of 32 items and set the second and the third items to True.

A CheckFeaturesAllowed action also allows you to use an integer to represent features to be verified. Each bit of the integer represents a feature. That is, 2^n represents the n^{th} feature, where n=0, 1, 2, …, 31. For verifying that both "Create New Customer" and "Modify Task" are activated then you need to use 6 to represent these 2 features, because $2^1 + 2^2 = 6$.

Create VerifyLicense Action

Right-click CopyProtector1, choose "Create Action". Choose "VerifyLicense":



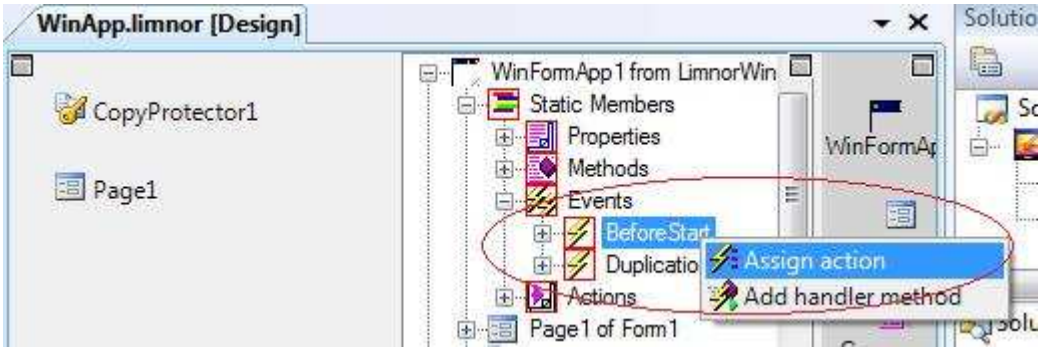
Verify License

We just created an action named **CopyProtector1.VerifyLicense**. When this action is executed, it verifies whether a license file exists. If a valid license exists then it finishes and does nothing. If a valid license does not exist then it shows a user interface for generating an activation file. The UI shows the contents of SupportMessage property. The UI asks for a serial number for the software. The UI is displayed only if the software allows user interaction. For software without UI, for example, a Windows Service, no UI is displayed. The software shuts down if no license is found.

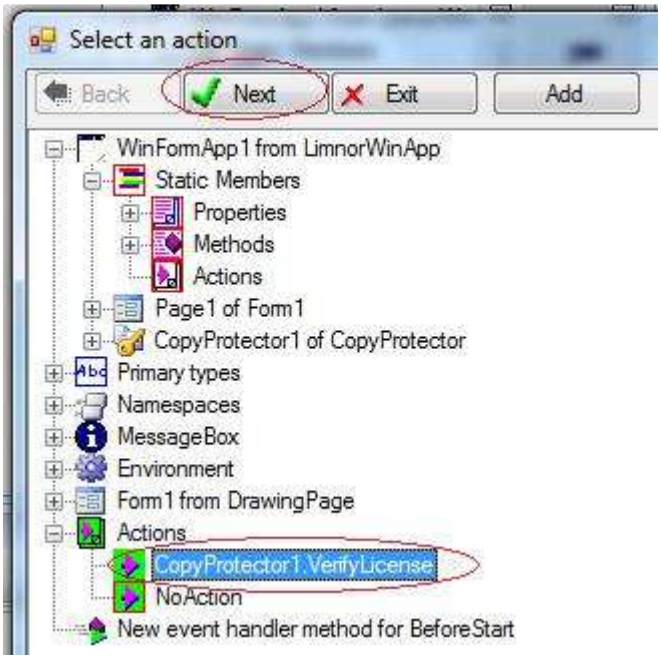
Now the question is that when should we execute this action. In term of Limnor Studio programming, the question is that which events we should assign this action to.

Usually we may execute this action at the beginning of the software. In a Windows Form application, **BeforeStart** can be the event to use.

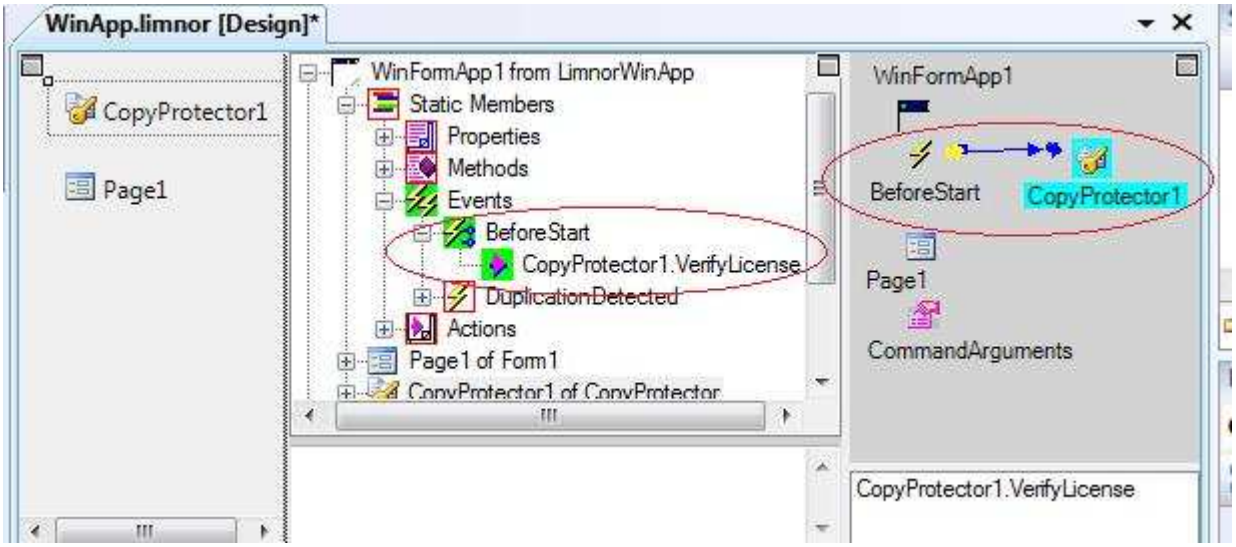
Right-click event BeforeStart, choose “Assign action”:



Select action **CopyProtector1.VerifyLicense**, click Next:



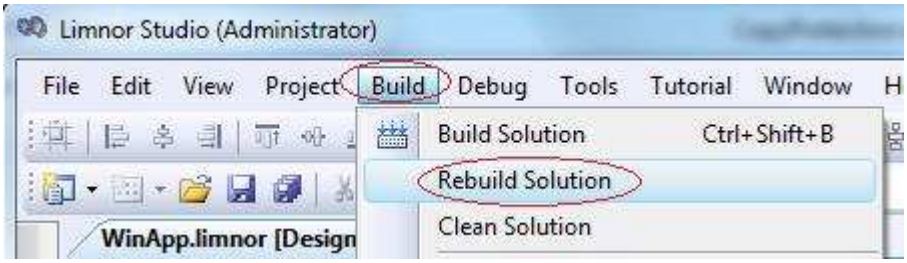
The action is assigned to the event:



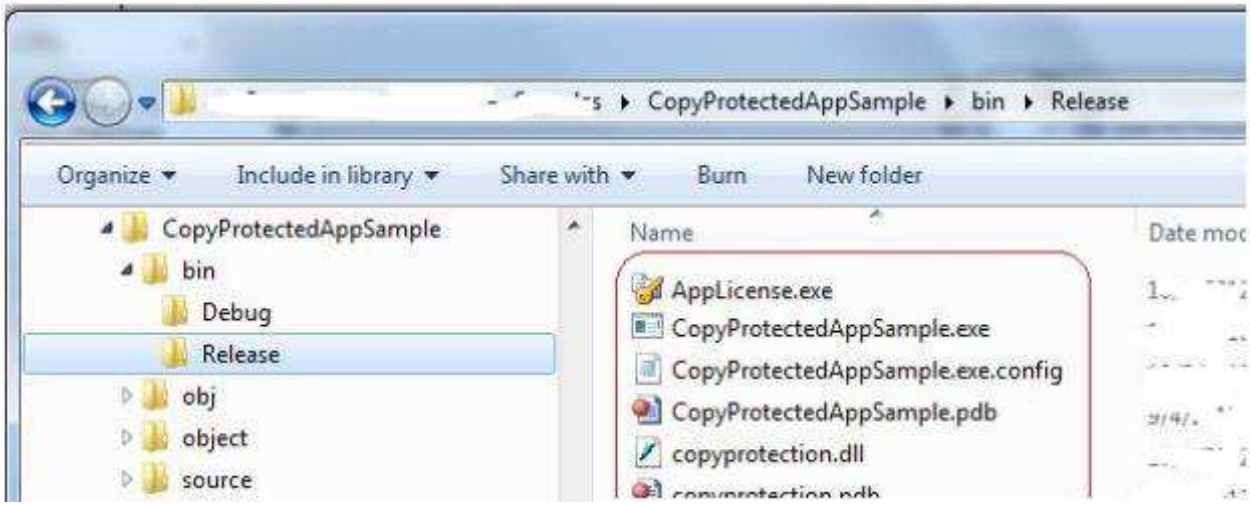
License Activation

Compile and Distribute Software

Build the project:



CopyProtectedAppSample.exe is the software generated. Distribute it together with the support file copyprotection.dll to users:

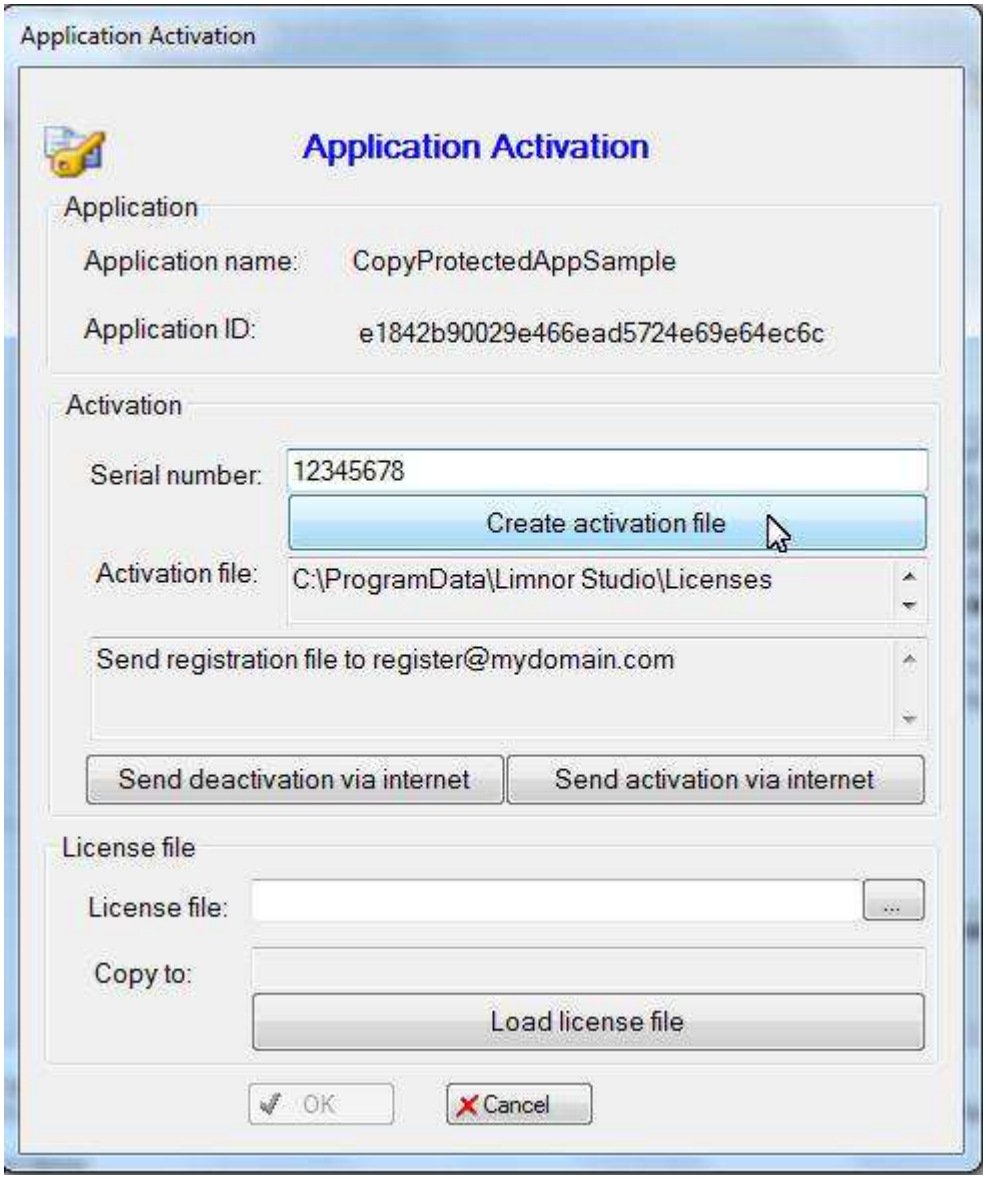


Create activation file

When the user runs CopyProtectedAppSample.exe, because it is not licensed, a license activation dialogue box appears. It allows the user to enter a serial number for the software. When the user purchases a license from you, you use a unique serial number to identify the license and give the serial number to the user.

Note that the contents of the property SupportMessage are displayed in the dialogue box.

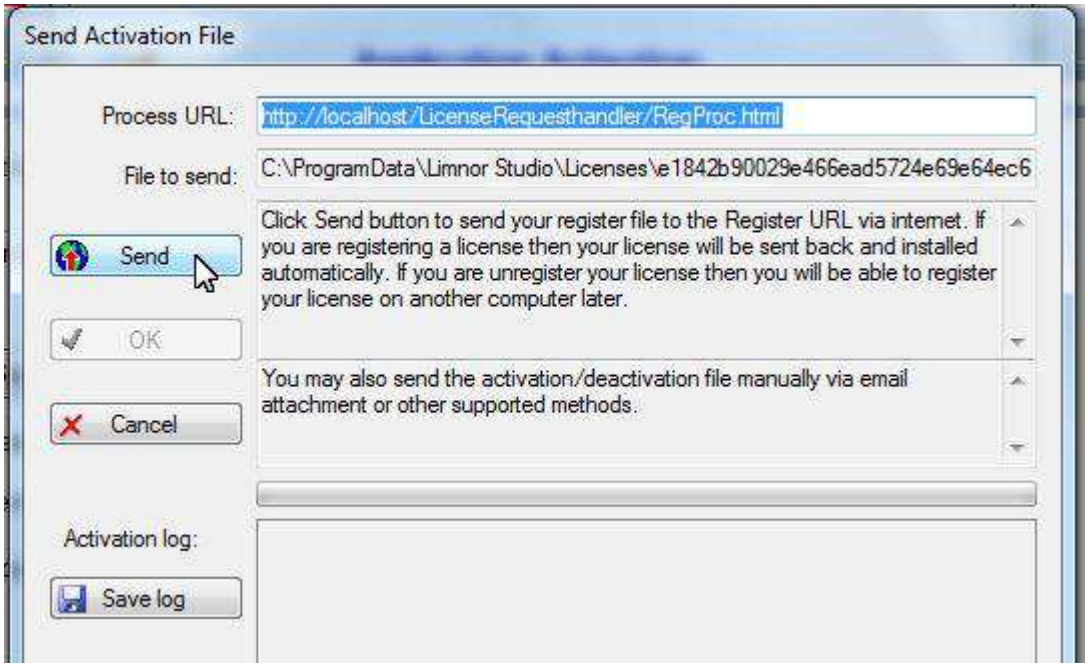
The user enters the serial number and click button “Create activation file”.



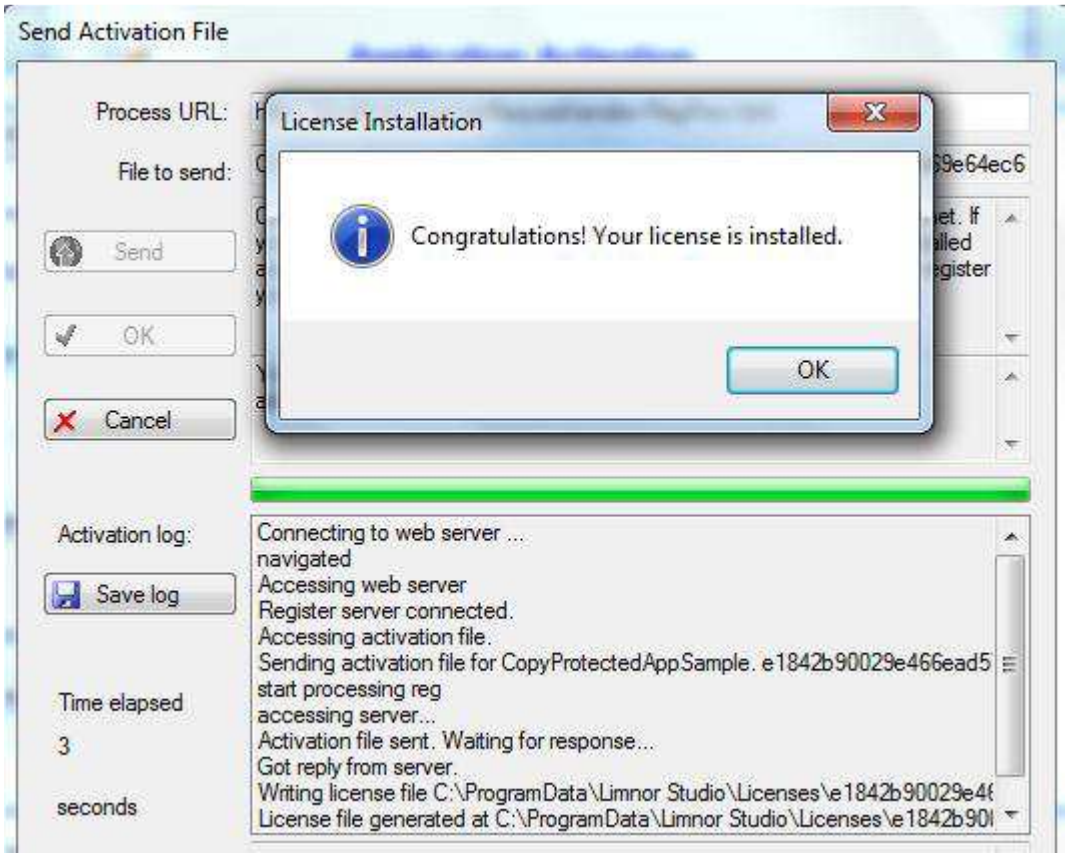
An activation file is created and a dialogue box appears for sending the file over the internet.

Automated Activation Processing

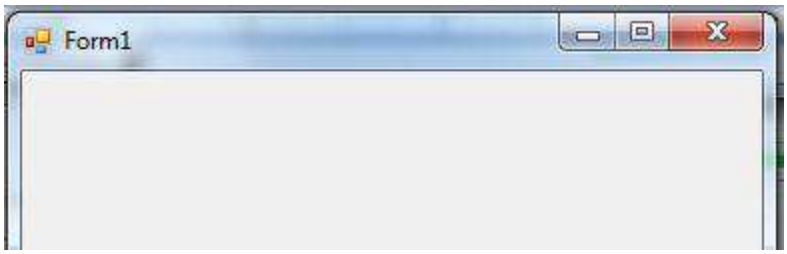
The user clicks “Send” button to send the file:



The processing of the activation file shows in the list box. If there are errors then the user checks the list to see the cause of problems. If the process finishes without an error then a message box appears notifying the success:



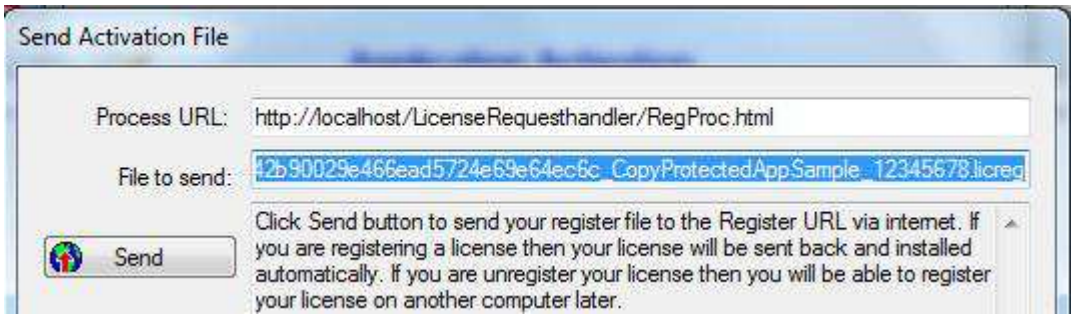
Click OK. The program gets license to run normally:



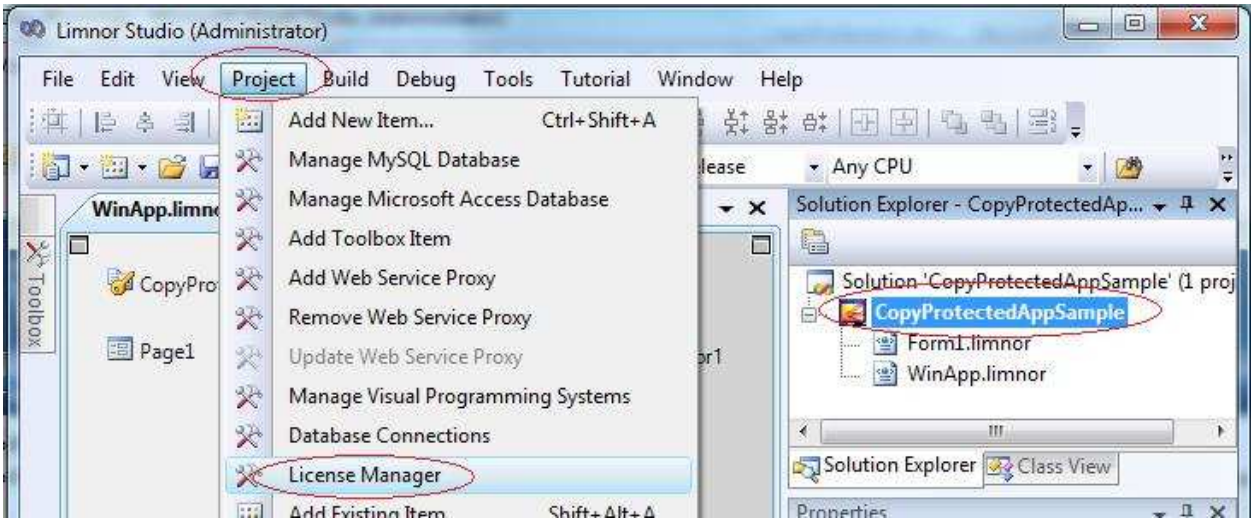
Manually Process Activation File

If you do not provide a web server for automated processing then the user sends the activation file to you manually, for example, via email attachment. You use Limnor Studio to process the activation file and create license file.

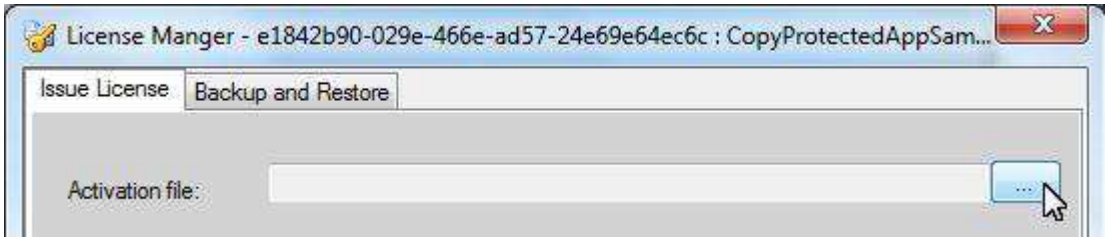
Your user locates full path of the activation file in the text box:



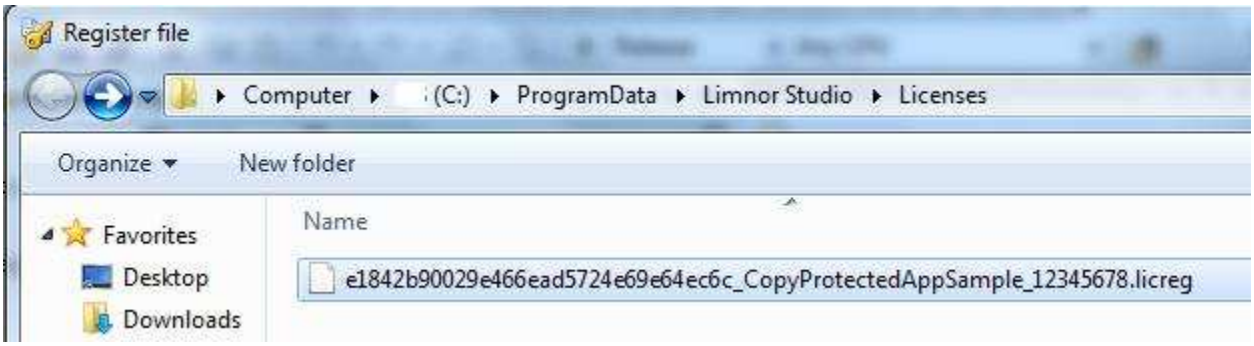
When you get the activation file from the user, open Limnor Studio and load the project for the software. With the project being the current project, choose menu “License Manager”:



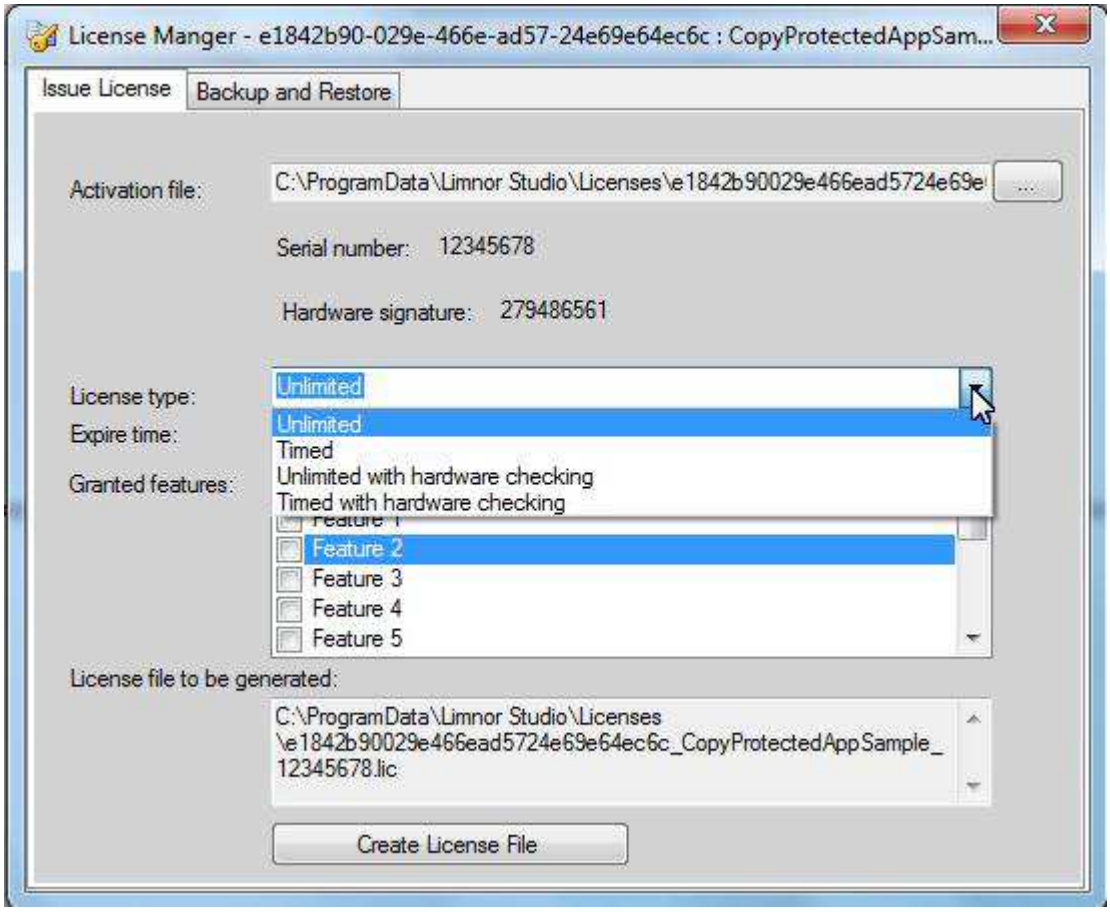
Click button  to load the activation file:



Open the activation file from the user:



If the activation file is valid then its contents are displayed:

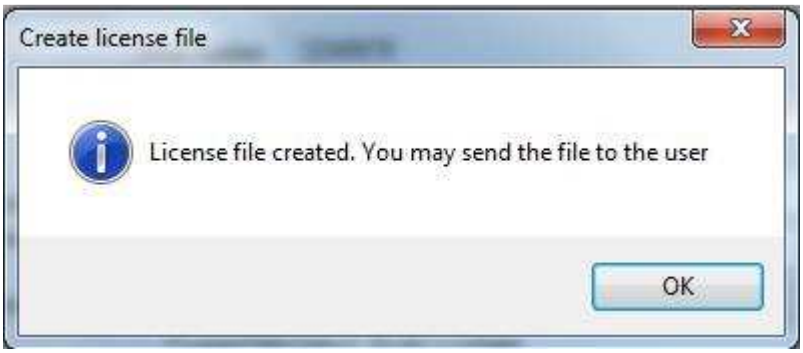


You use the serial number and hardware signature to help you to decide whether you want to grant a license to the user. Hardware signature is a hash of hard disk serial numbers and network card addresses. It uses a hash to protect the privacy of your user. When two activation files are using the same serial number, hardware signature helps you to determine if these two activations are for the same computer. But note that hardware may change for the same computer.

License type:

- Unlimited – the license will not expire
- Timed – the license will expire on the day specified by "Expire time"
- Unlimited with hardware checking – the license will not expire. Hardware signature will be verified when VerifyLicense action is executed.
- Timed with hardware checking – the license will expire on the day specified by "Expire time". Hardware signature will be verified when VerifyLicense action is executed.

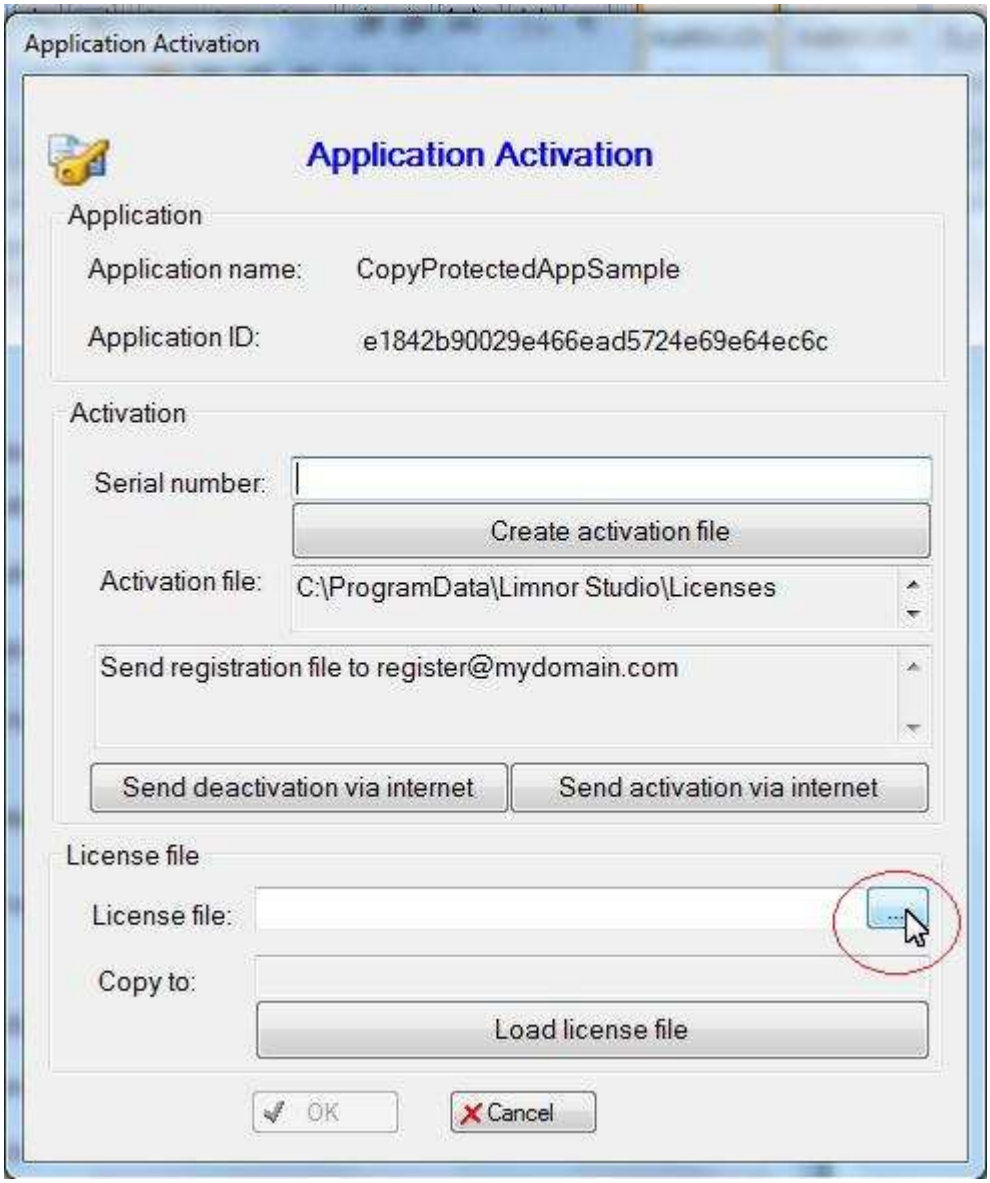
Click button "Create License File" to generate a license file.



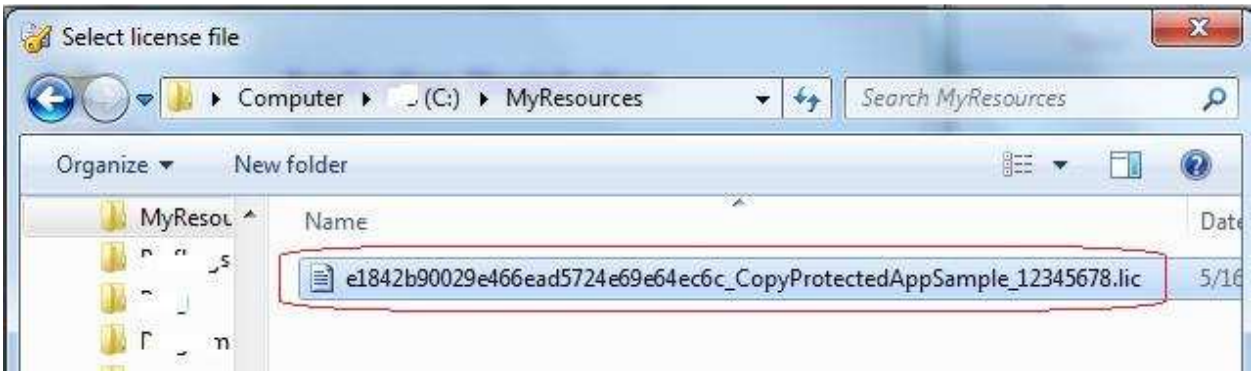
Note that issuing licenses from Limnor Studio IDE does not connect to your license manager plug-in. So, there is no record in your license management database.

Manually Install License File

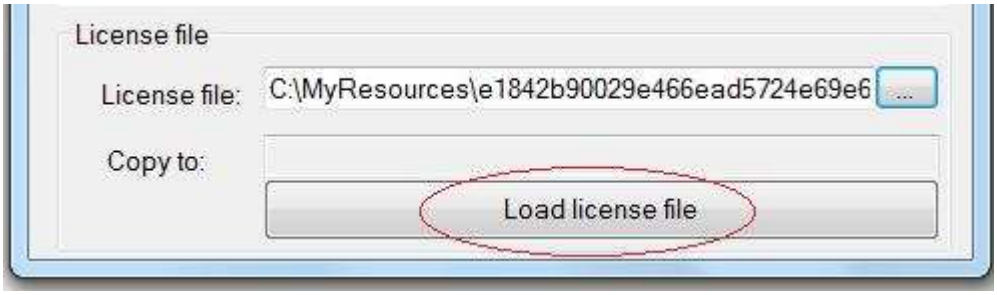
You may send the license file to your user via email attachment. The license file should be copied to folder {ProgramData}\Limnor Studio\Licenses for it to be used. If the user uses Windows 7 in C: drive then the folder is C:\ProgramData\Limnor Studio\Licenses. The activation dialogue box helps your user to do the copy. Run the software, because the software is not licensed, the activation dialogue box appears. Click button  to find the license file:



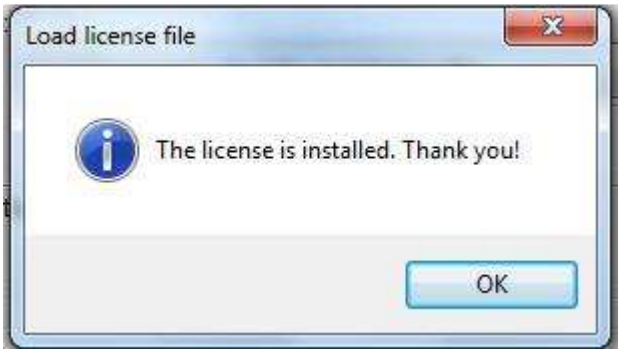
Open the license file:



Click button “Load license file”:



If the license file is valid then it is copied to the right folder and the user is notified:

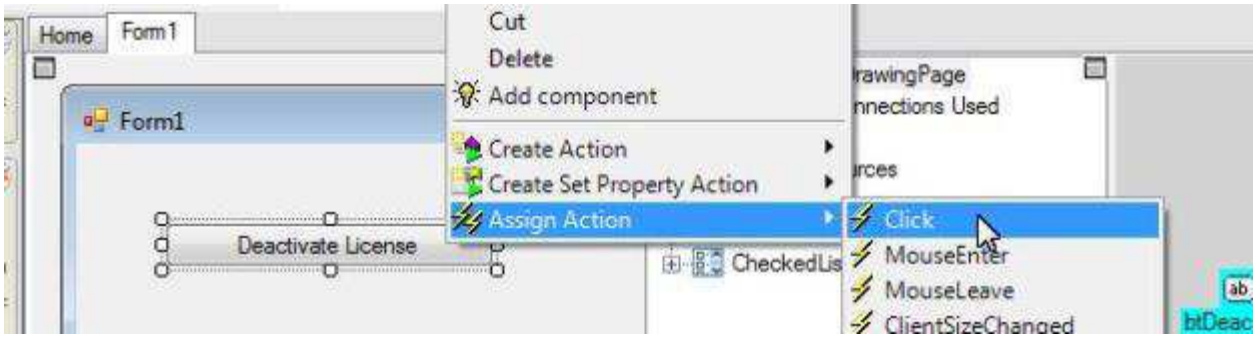


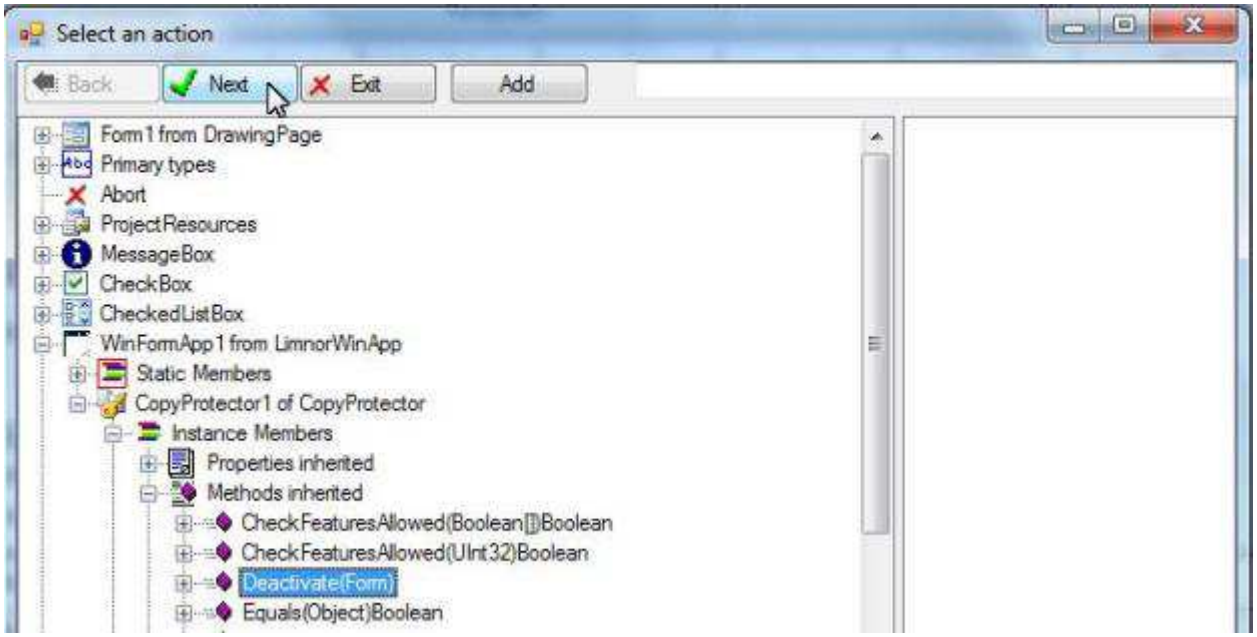
From now on the software runs normally.

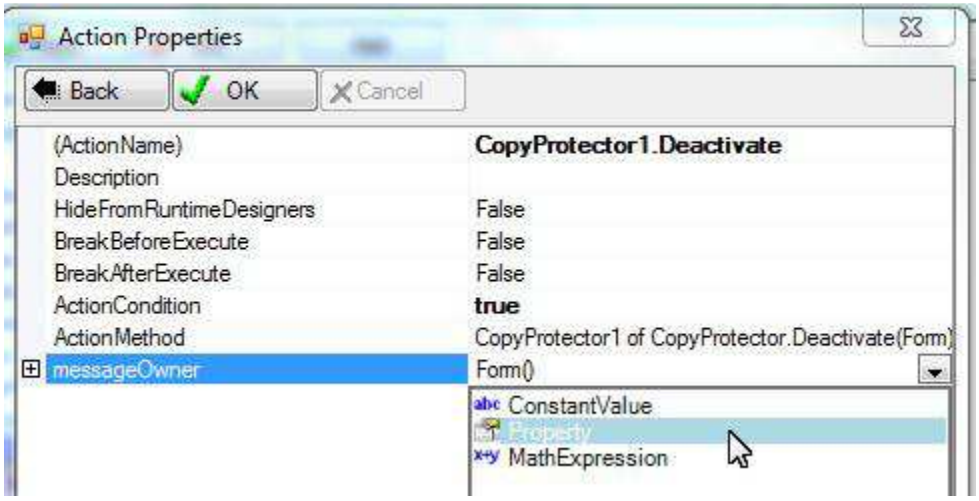
License Deactivation

If you want to allow your users to deactivate licenses then you may provide a chance for your program to execute a Deactivate action. Once a license is deactivated from a computer it can be re-activated on any supported computer again.

Suppose we use a button to execute a Deactivate action:

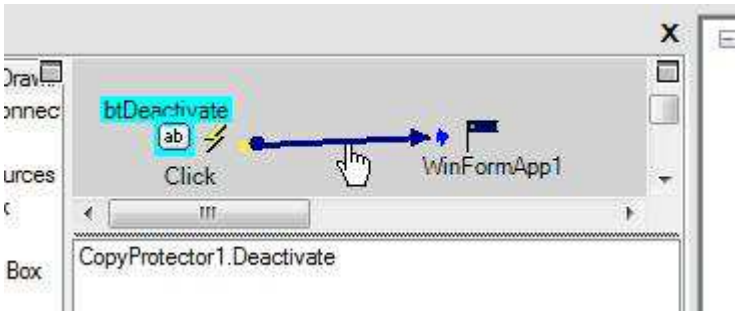



Select an action
Back
Next
Exit
Add
Form1 from DrawingPage
Primary types
Abort
ProjectResources
MessageBox
CheckBox
CheckedListBox
WinFormApp1 from LimnorWinApp
Static Members
CopyProtector1 of CopyProtector
Instance Members
Properties inherited
Methods inherited
CheckFeaturesAllowed(Boolean[])Boolean
CheckFeaturesAllowed(UInt32)Boolean
Deactivate(Form)
Equals(Object)Boolean

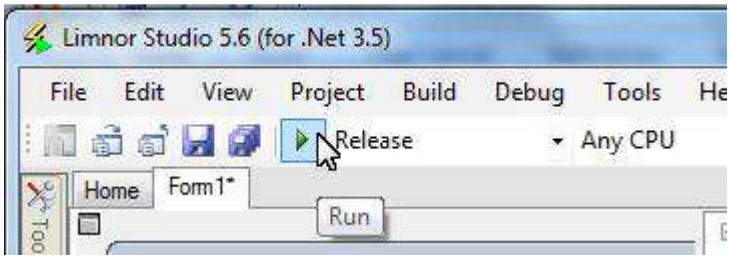

Action Properties
Back
OK
Cancel
(ActionName)
CopyProtector1.Deactivate
Description
HideFromRuntimeDesigners
False
BreakBeforeExecute
False
BreakAfterExecute
False
ActionCondition
true
ActionMethod
CopyProtector1 of CopyProtector.Deactivate(Form)
messageOwner
Form()
ConstantValue
Property
MathExpression


Object picker
Back
Next
Exit
Add
Form1 from DrawingPage
Primary types
ProjectResources

An action is created and assigned to the button:

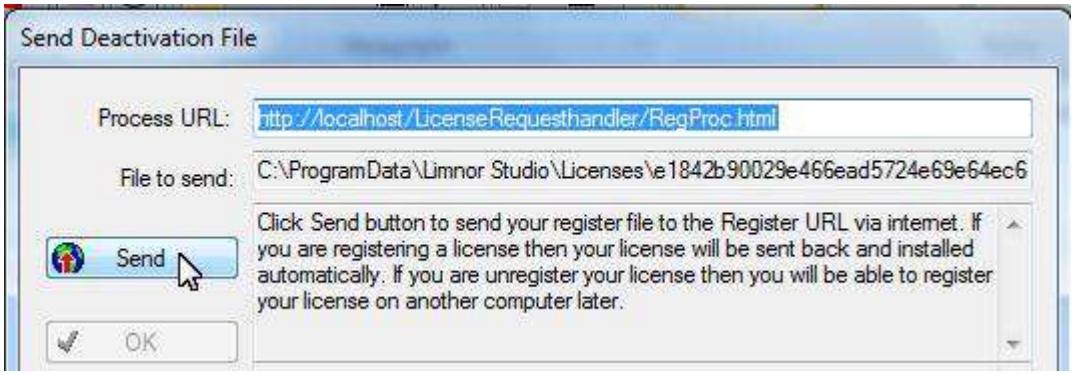


Run the program again:

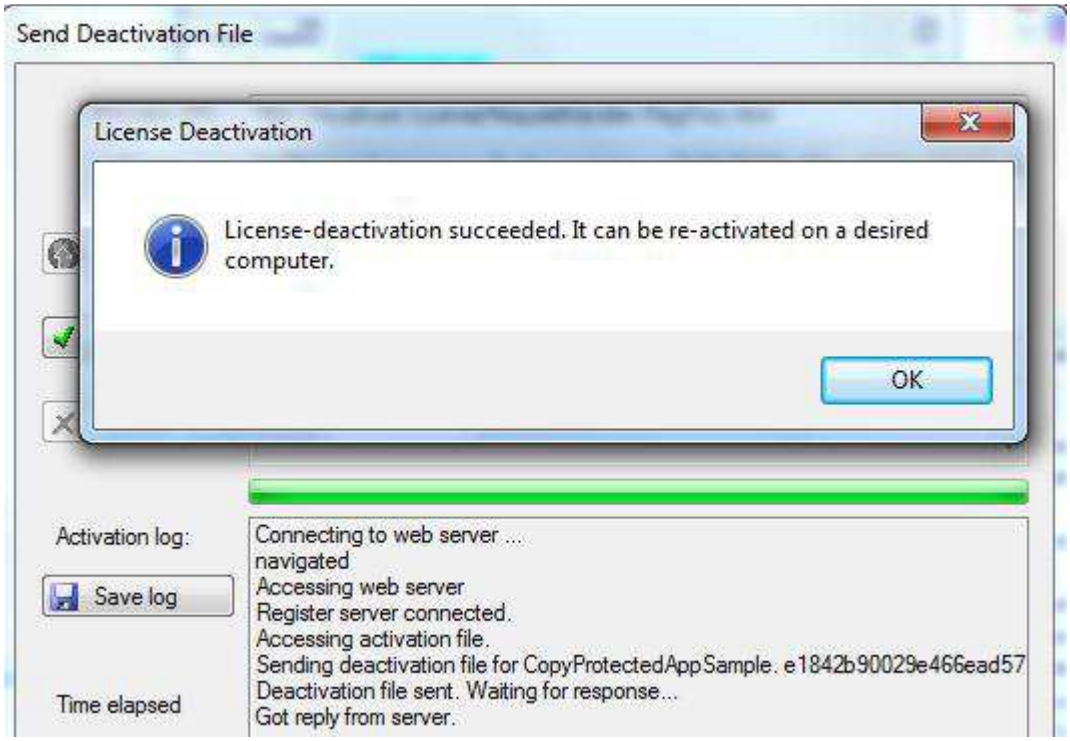


Click the button:

A dialogue box appears for sending the deactivation file to the web server. Click “Send”:



A message box appears if the processing finishes without an error:



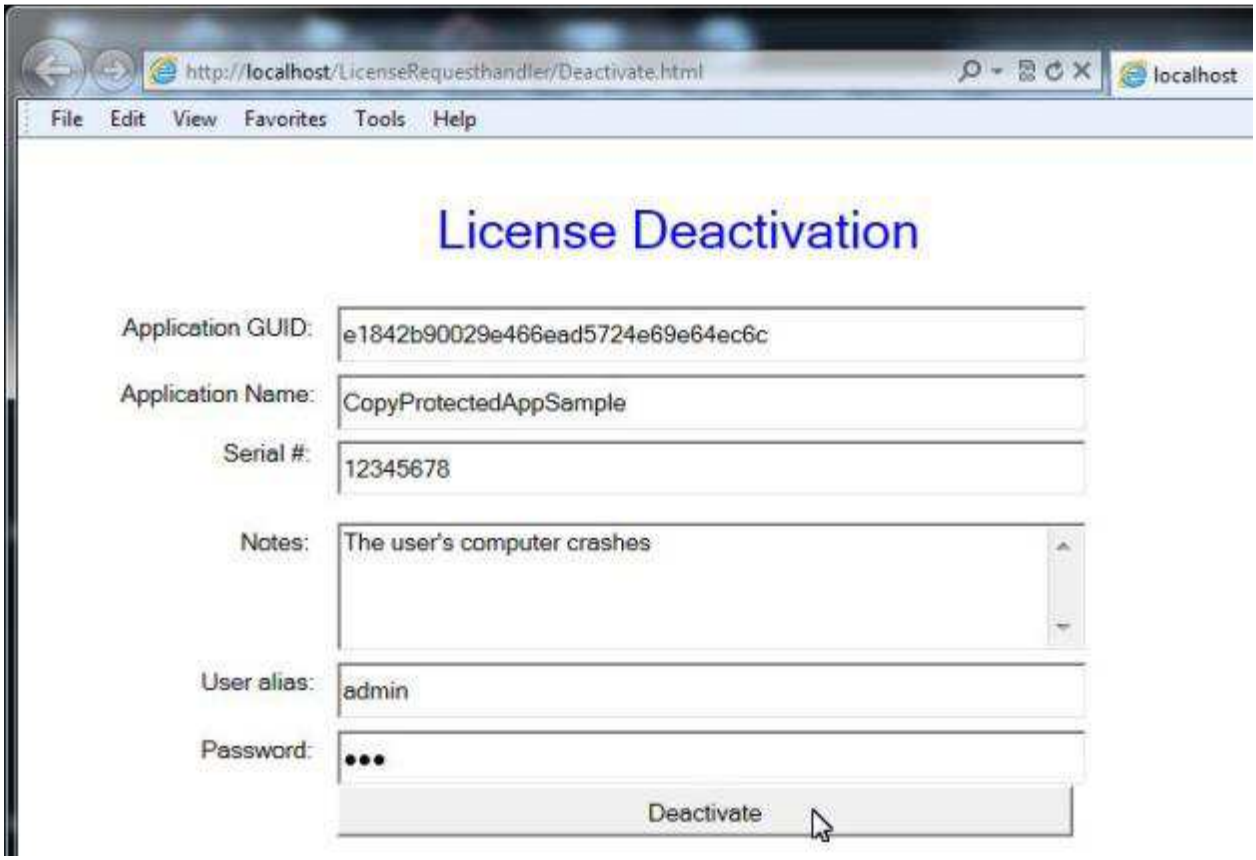
Manual License Deactivation

License deactivation may not always be done before a user needs to re-activate a license. For example, a computer hard disk failure, or the user simply forget it before re-formatting hard disk. In such situations your license management plug-in blocks license re-activation because it thinks that a license is already been activated.

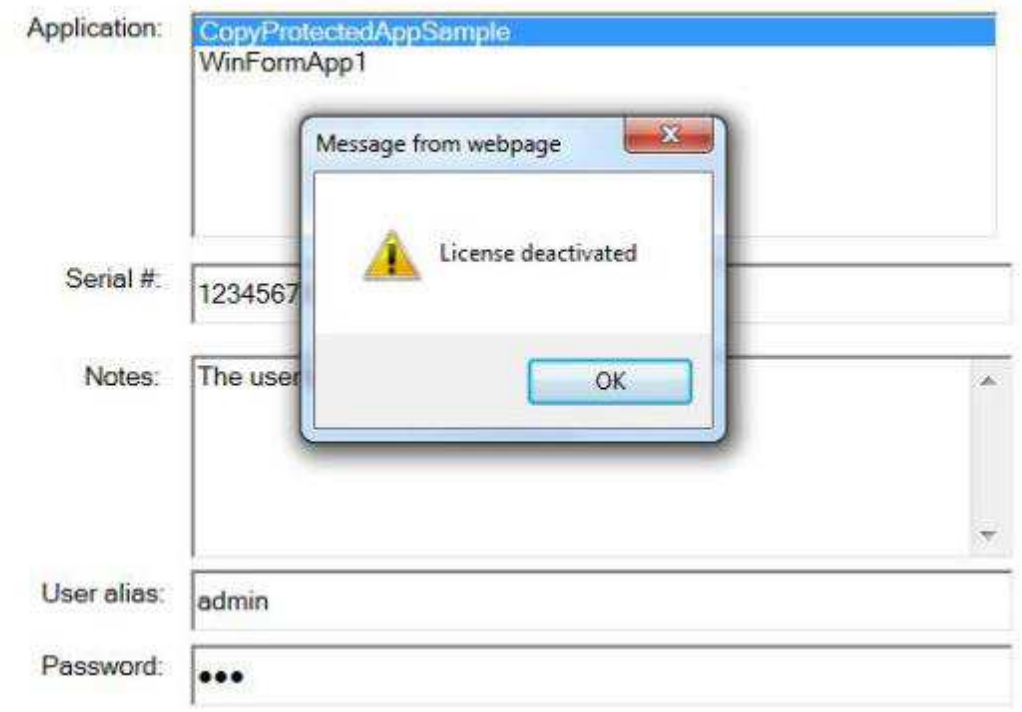
Web page deactivate.html allows you to force deactivate a license. Suppose you setup your web server as http://www.mydomain.com/LicenseRequesthandler

then the URL is http://www.mydomian.com/LicenseRequesthandler/deactivate.html

Launch the deactivation URL in a web browser. Enter application GUID, application name, a serial number and other information. Click “Deactivate”:



A message box appears if the license is deactivated:



Now your user may activate his/her license from any supported computer.

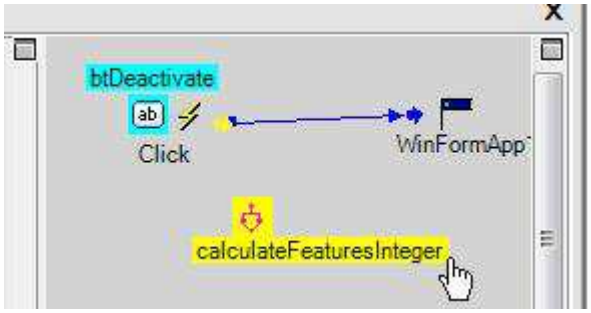
Verify Protected Features

An action of CheckFeaturesAllowed returns True if given features are allowed. You may program your software using the action return value.

We use an example to demonstrate the process of verifying protected features. We use a checked list box to display 32 items, representing 32 protected features. The user may check items, indicating features to be checked. When the user clicks a button, an integer representing the features to be checked is calculated and displayed; the integer is also used in a CheckFeaturesAllowed action; a message box appears showing the result of the action.

In your applications you would rarely let the user select features to be checked. Usually you just use pre-calculated integers. This sample can be a utility to give you integer values for any features combinations you want to verify. This sample project can be downloaded from
http://www.limnor.com/studio/CopyProtection.zip

We create a method named calculateFeaturesInteger. It returns an integer corresponding to selected check list items.



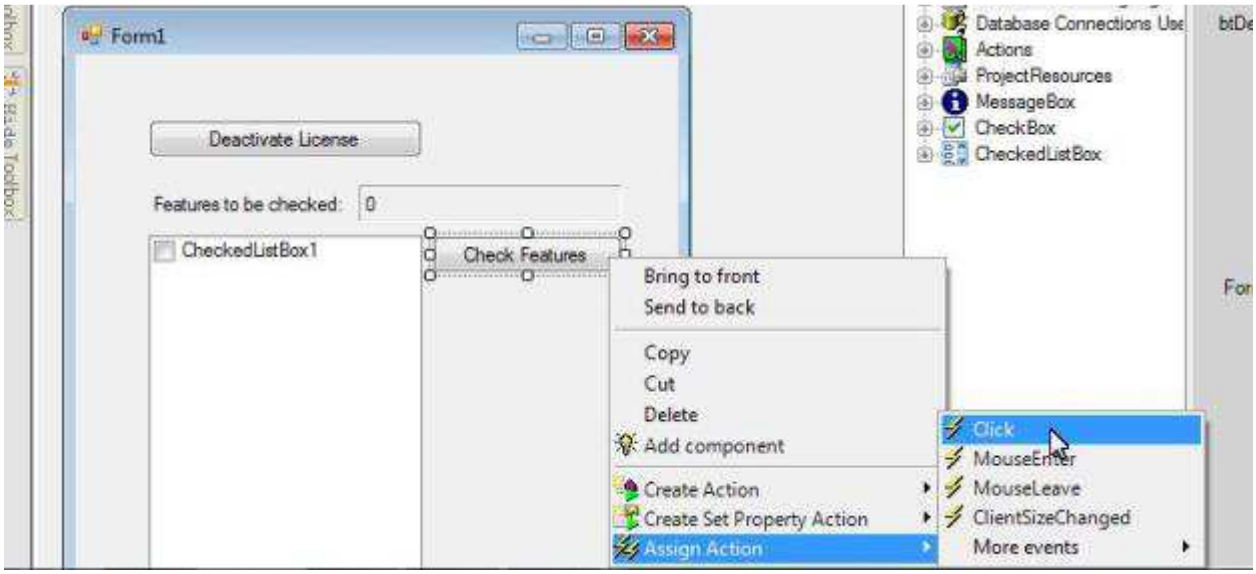
Sample of verifying protected features

Choose verification events

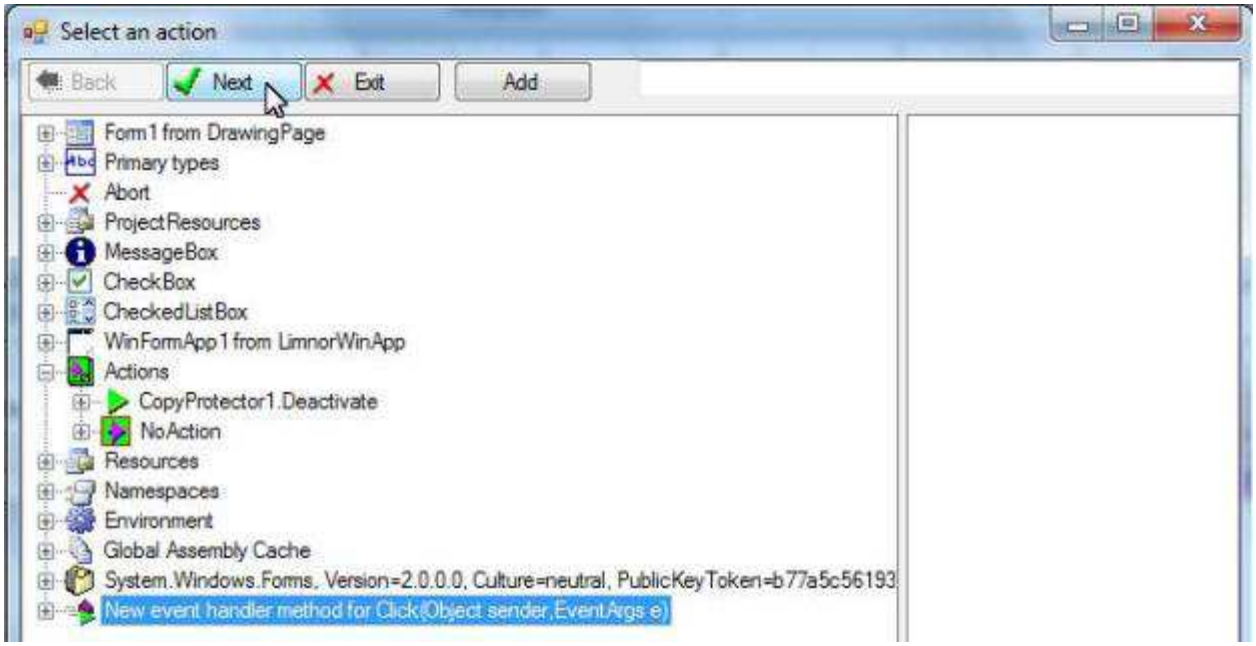
In your programs, you need to choose at which events different feature combinations should be checked.

For this sample, we choose Click event of button “Check Features” as one event for checking features selected by the user on the checked list box.

On clicking button “Check Features”, we want to execute calculateFeaturesInteger to get an integer for verifying features. The integer is displayed on a label; the integer is used to execute a CheckFeaturesAllowed action. These programming are shown below.



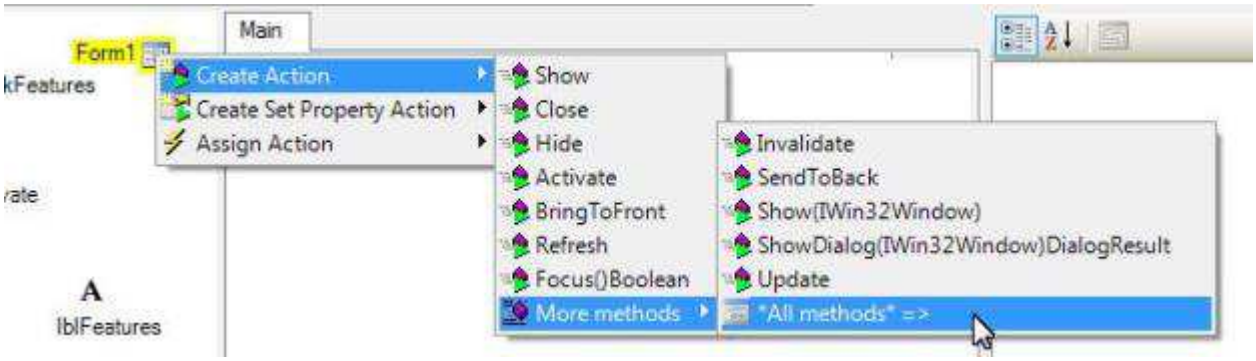
Create an event handler so that complex logic can be programmed:



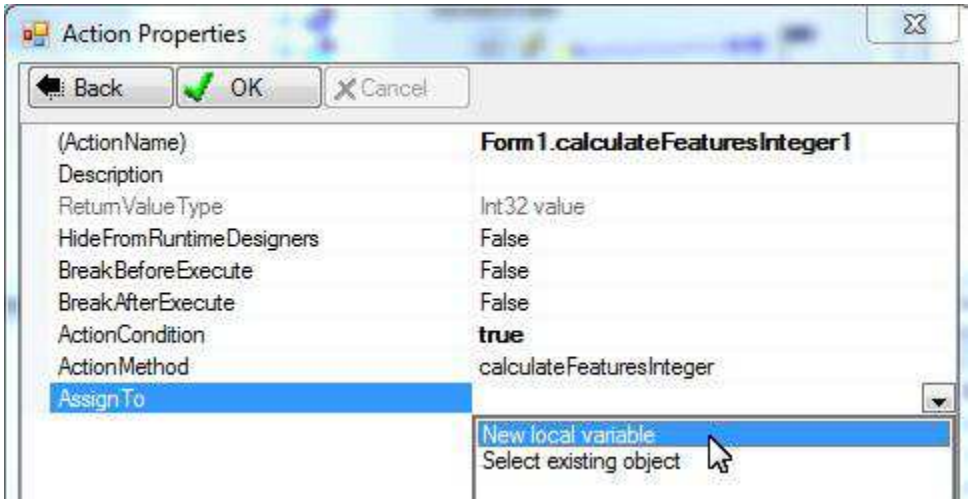
Determine signature integer for features

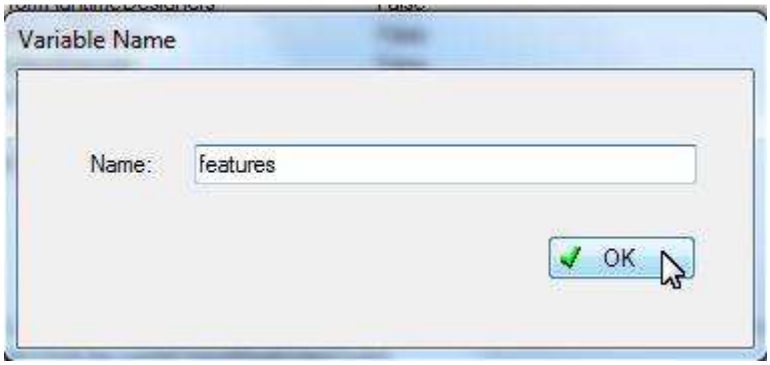
In your programs most likely you will use an integer constant to represent the features to be verified. You may use this sample as a utility to determine integer values for any feature combinations.

In this sample we use method calculateFeaturesInteger to dynamically calculate it based on the selections on the checked list box:

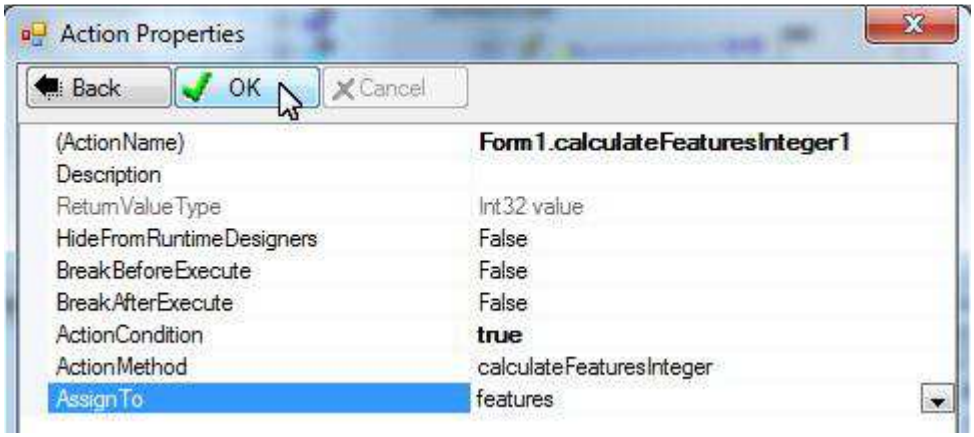


Select “New local variable” for “AssignTo” of the action:

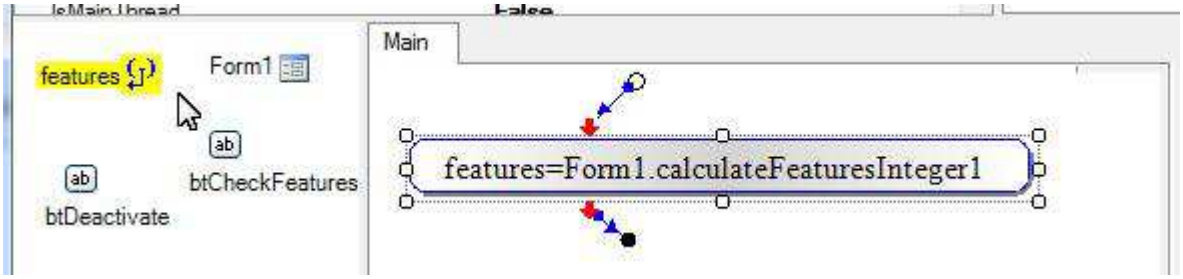




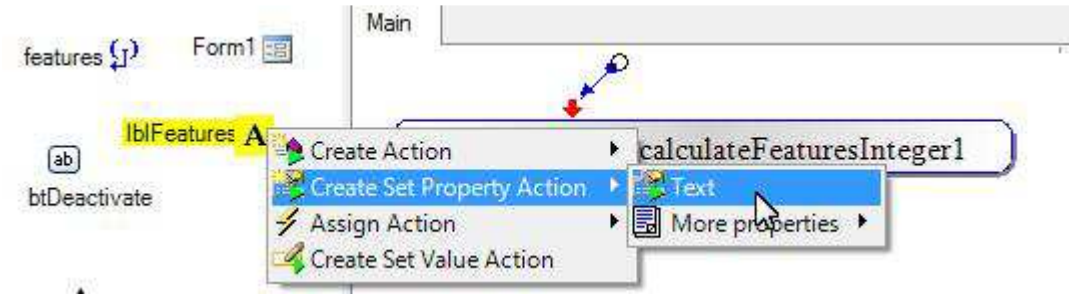
Click OK to finish creating this action:

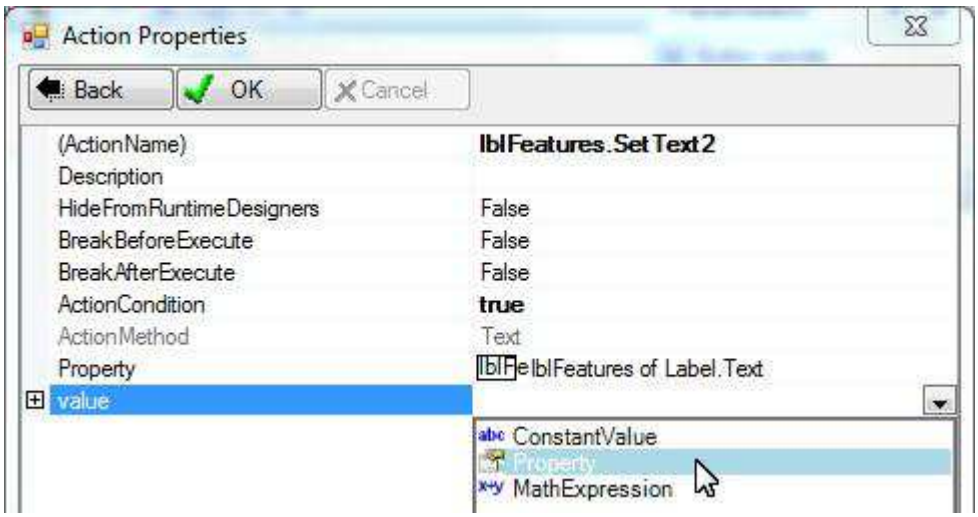


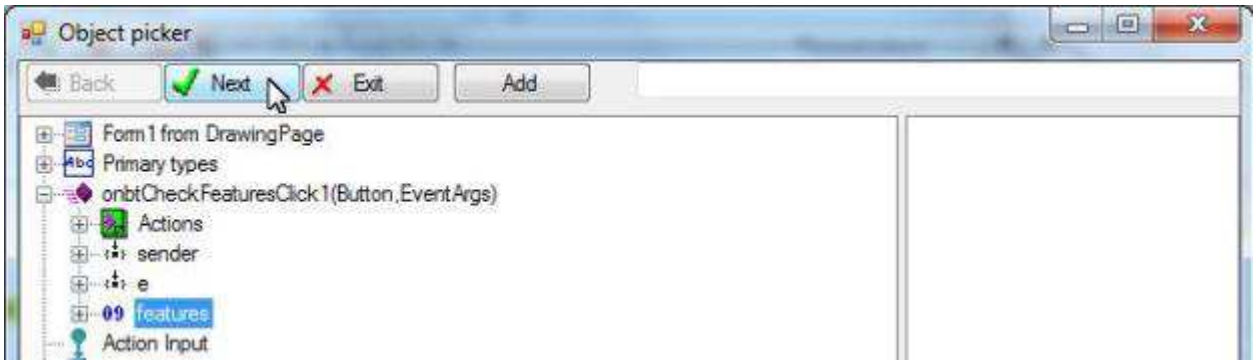
The action is created. A new variable named features appears in the Variable Pane.

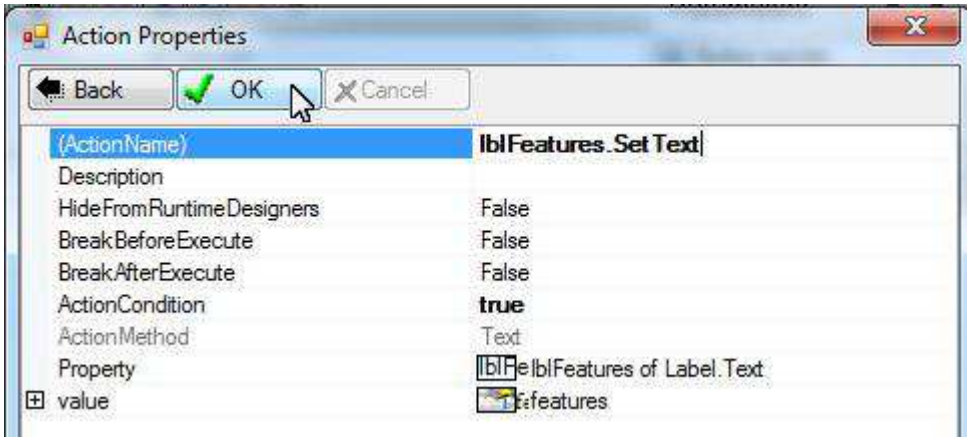


Show the variable on a label:

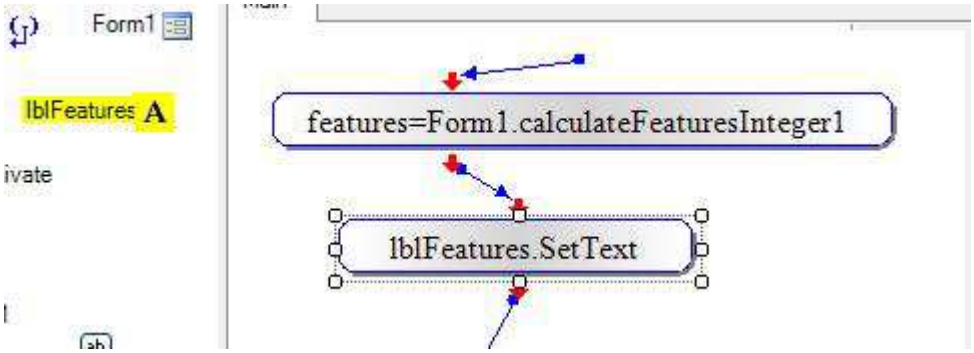






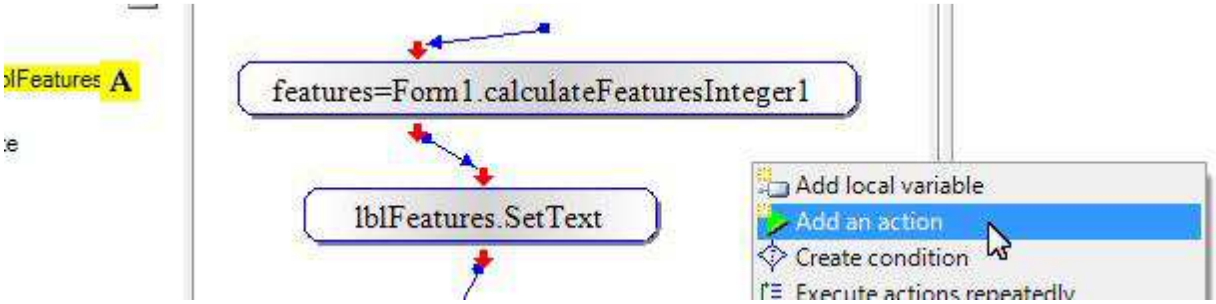


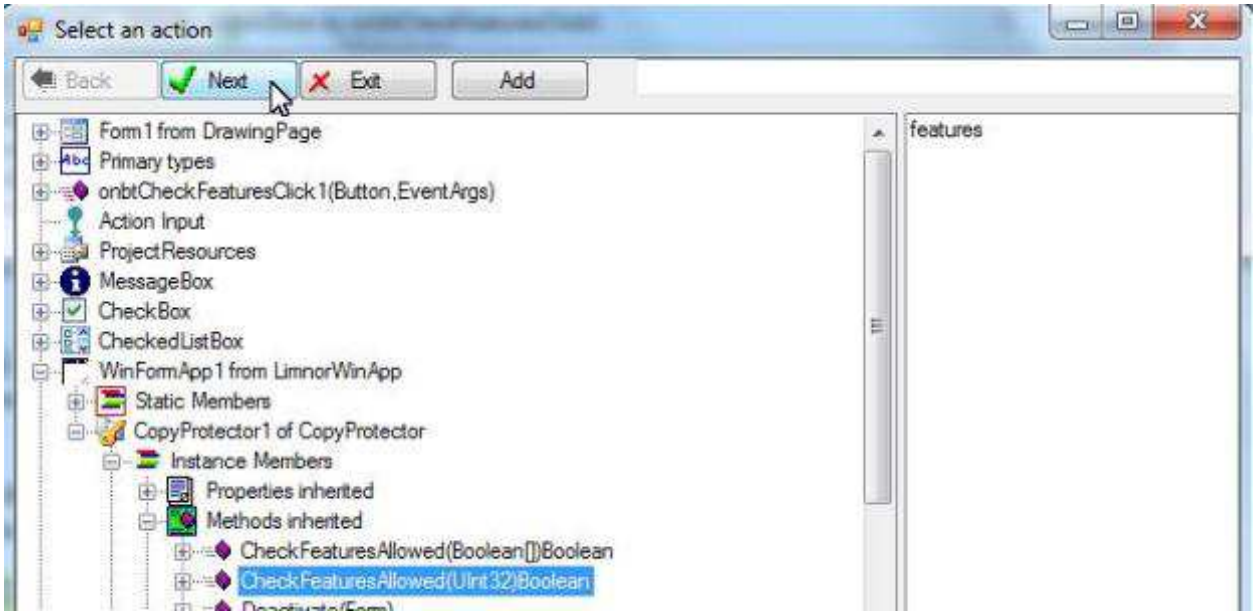
Link it to the last action:



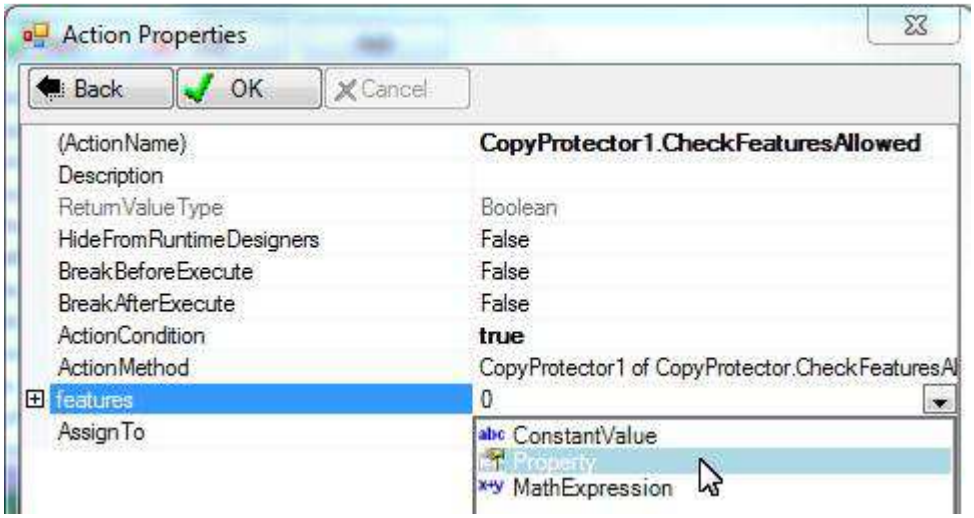
Create Verification Action

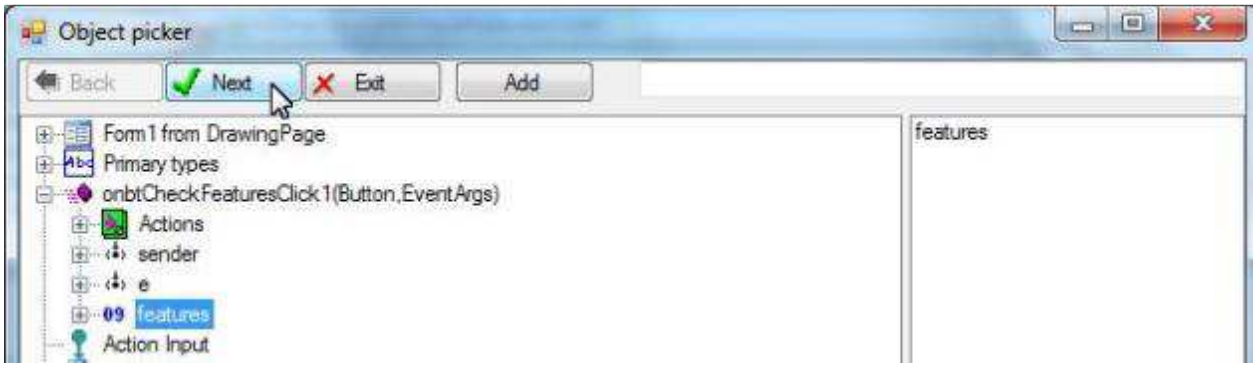
Now let's create an action of CheckFeaturesAllowed:



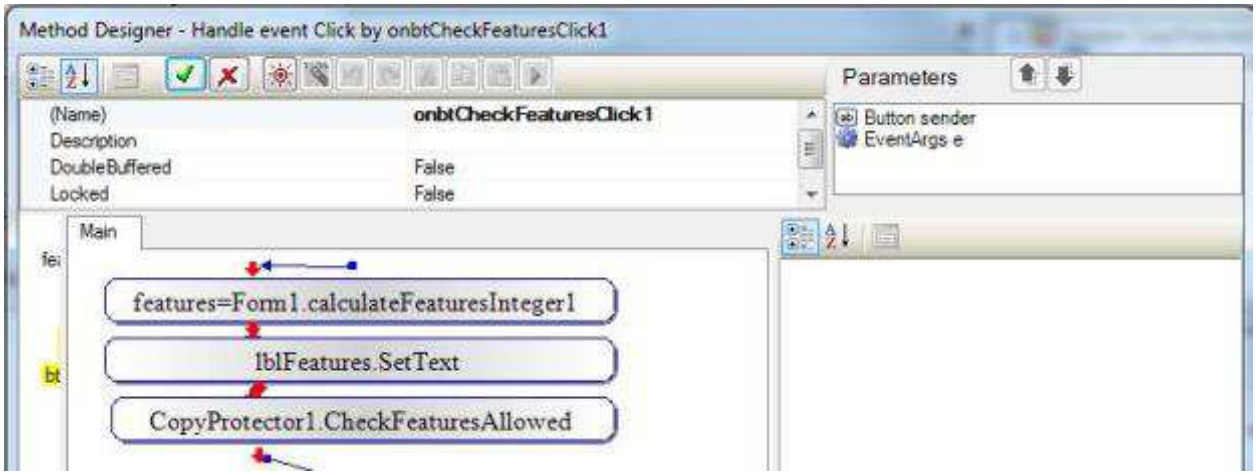


Pass the integer we got to the action:



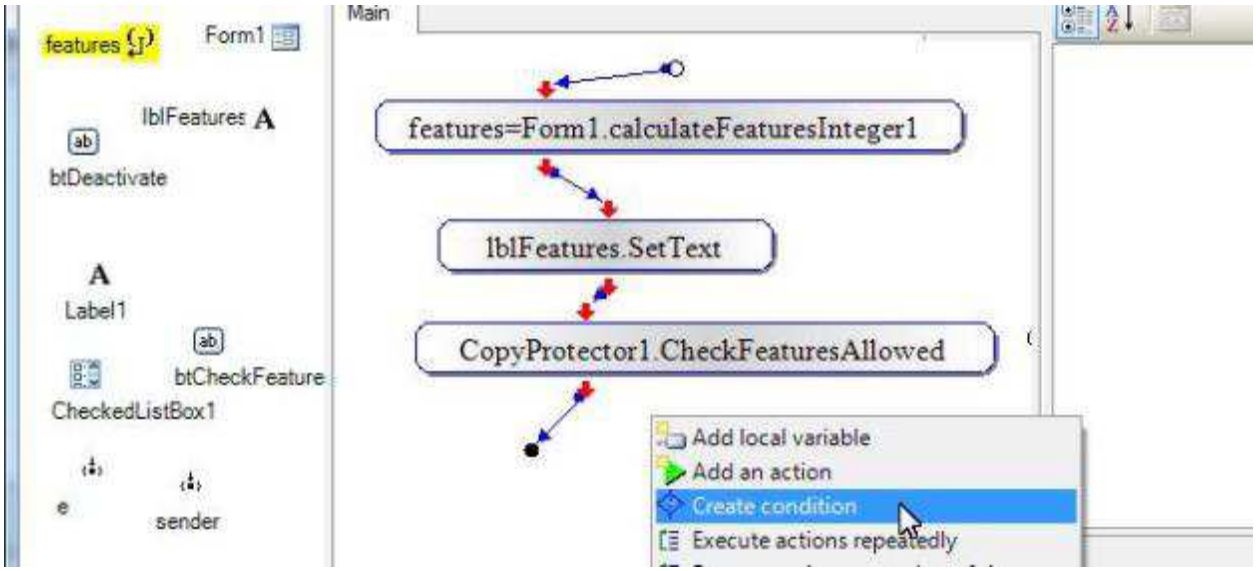


Click OK to finish creating this action. Link this action to the last action.

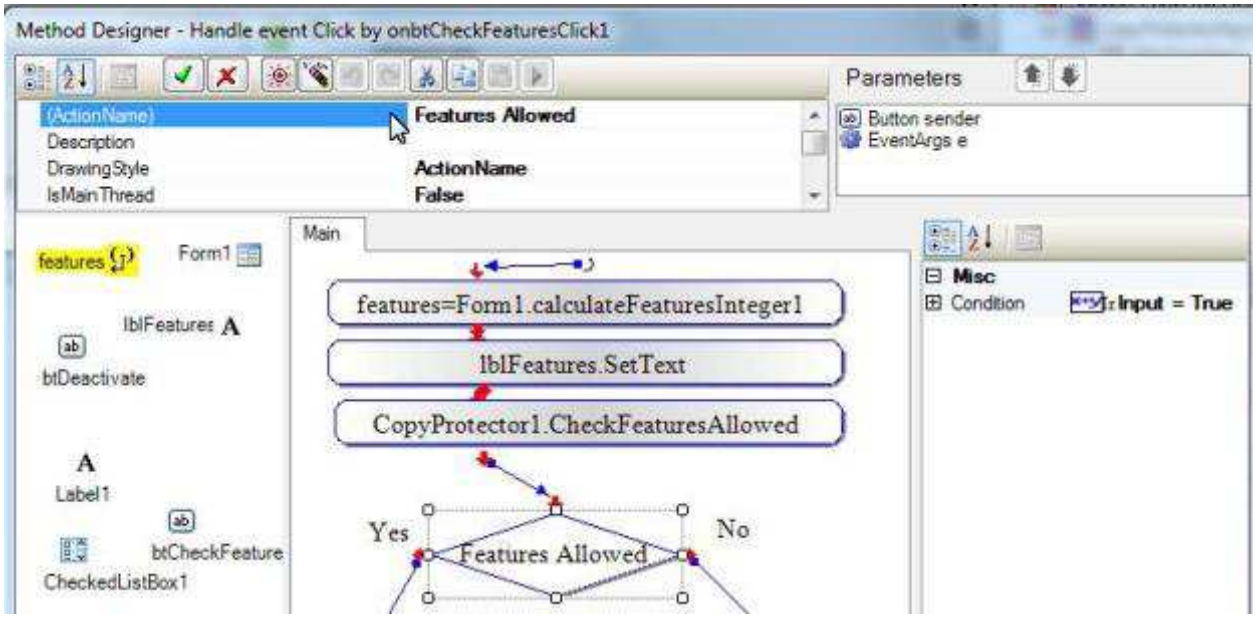


Use Verification Result

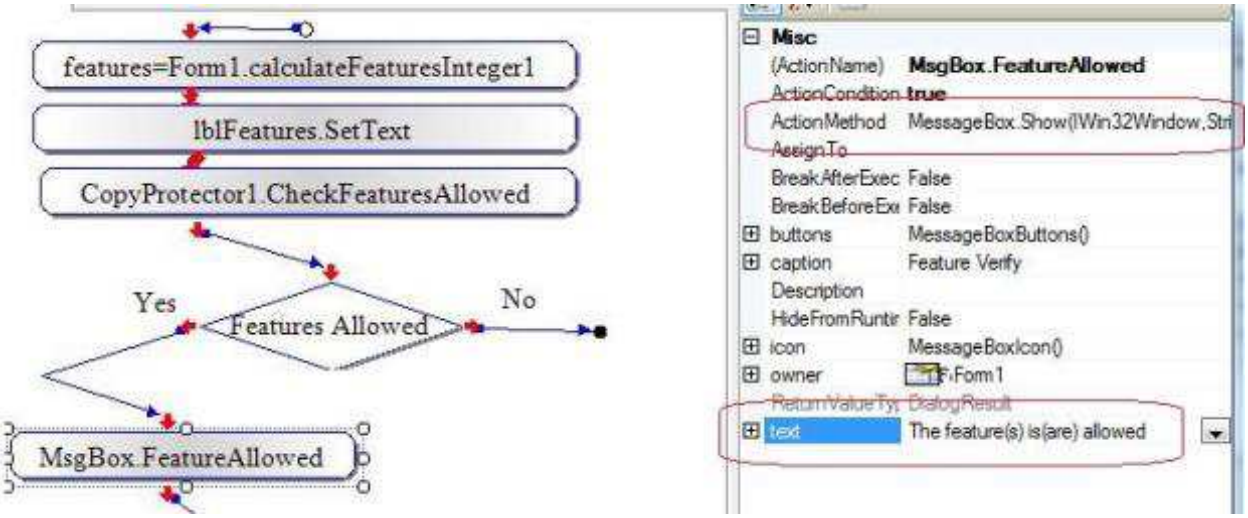
Let's create a Condition action for using the result of CheckFeaturesAllowed action:

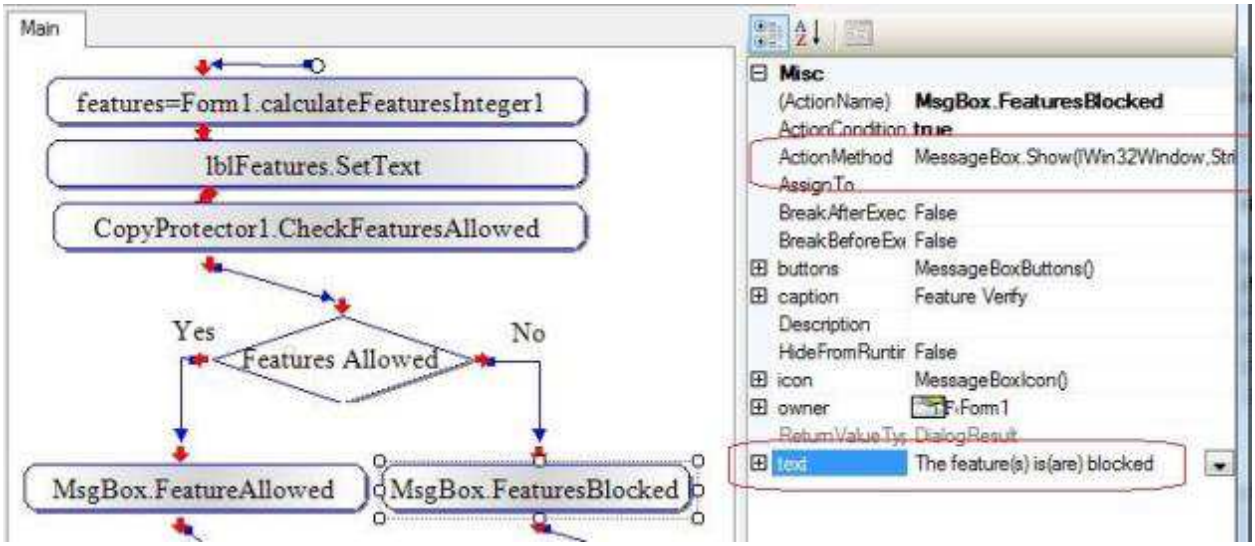


Link the Condition action to CheckFeaturesAllowed action. Rename the Condition to "Features Allowed":



How to respond to the result of CheckFeaturesAllowed depends on your software business logic. For this sample, we simply display a message box to show the result.





We may finish creating this event handler for button “Check features”:

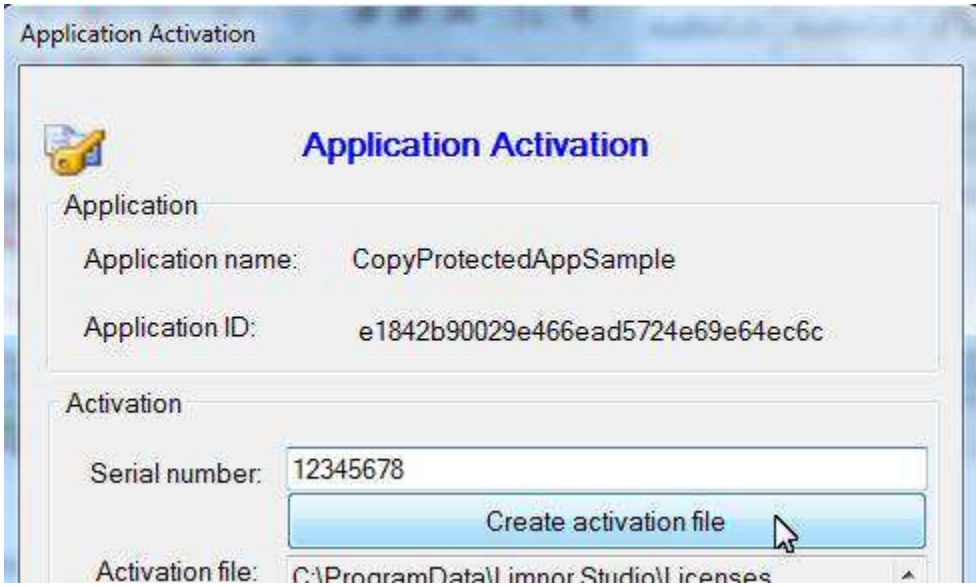
Granting Protected Features

Allowed features are issued with each license.

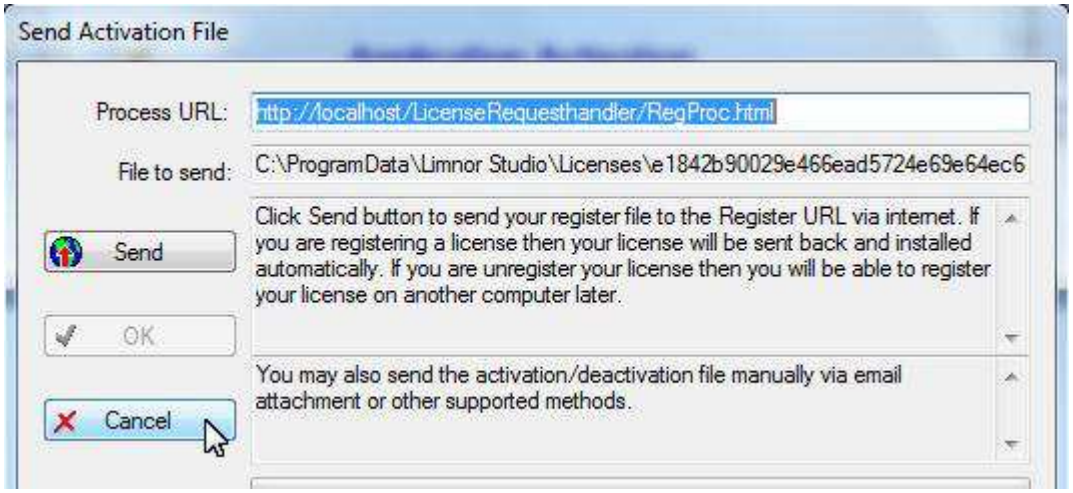
If you use automated license activation system then allowed features should be recorded for each serial number in your database.

If you manually issue licenses using Limnor Studio then you need to select allowed features at the time of creating license. Below we show such processes.

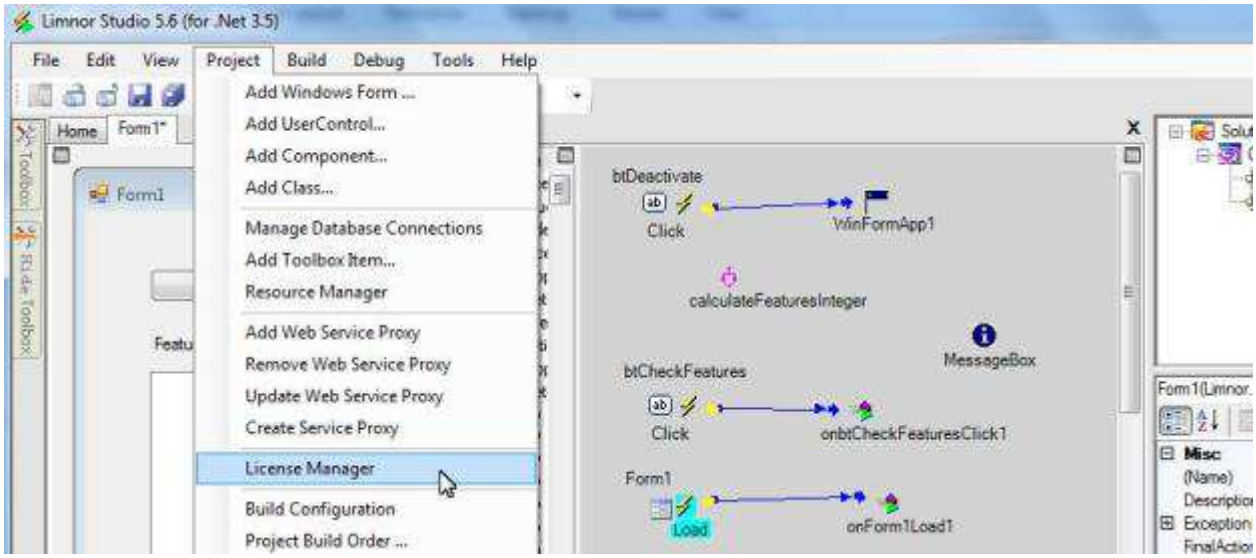
Deactivate the license first. Then run the program. It will ask for serial number and create an activation file.



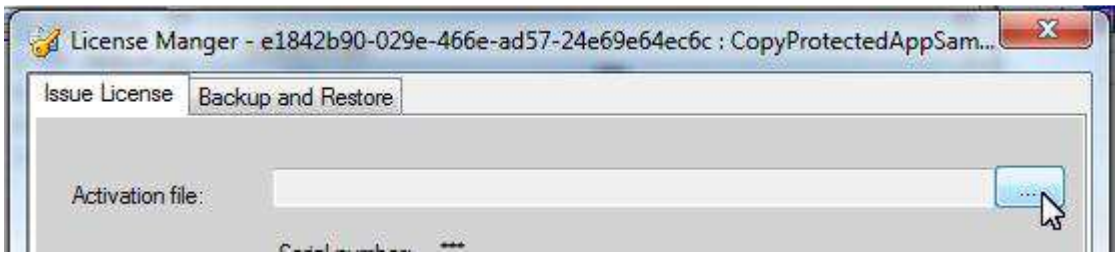
Cancel the sending of the activation file because we want to manually process it with Limnor Studio:

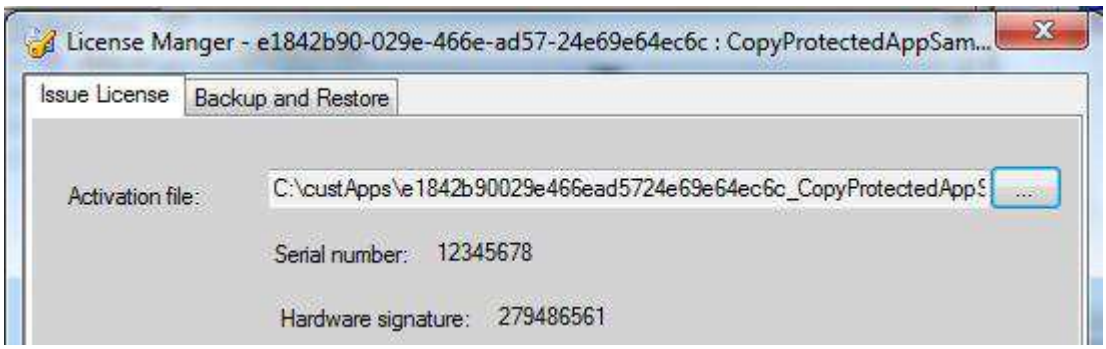


Open License Manager for the project:

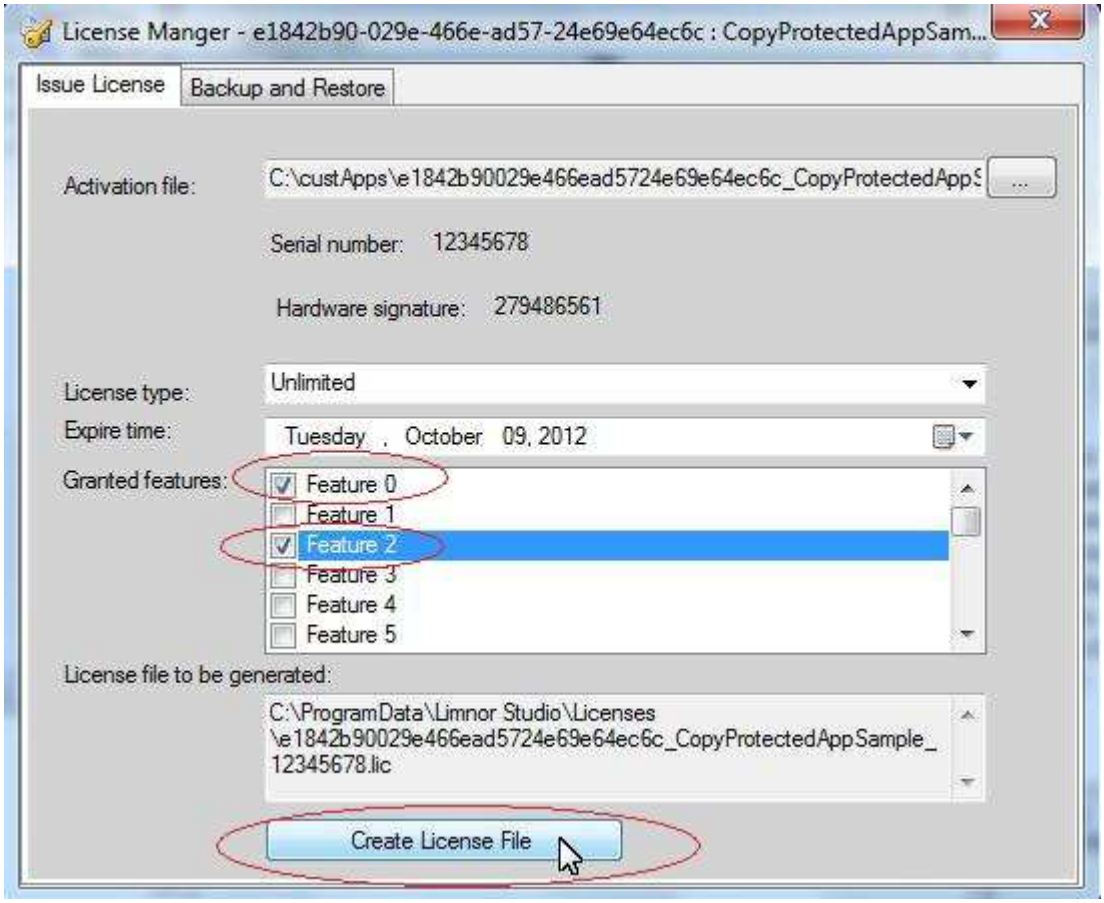


Load the activation file:



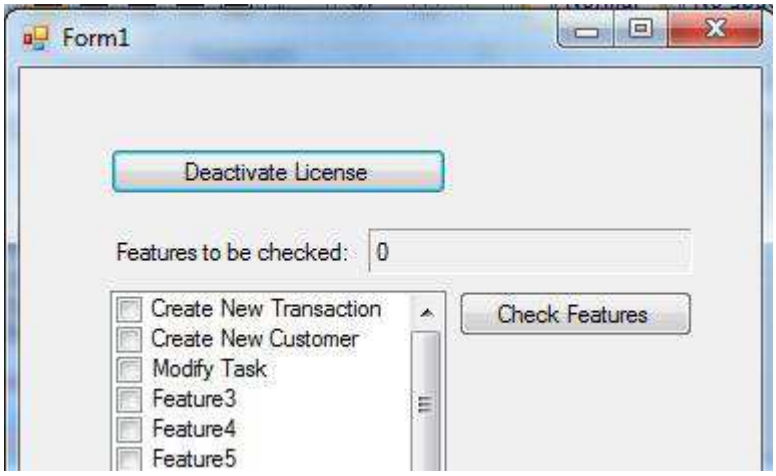


Before we create a license file for this activation file, we may choose which features to allow. Suppose we allow the first and the third features:

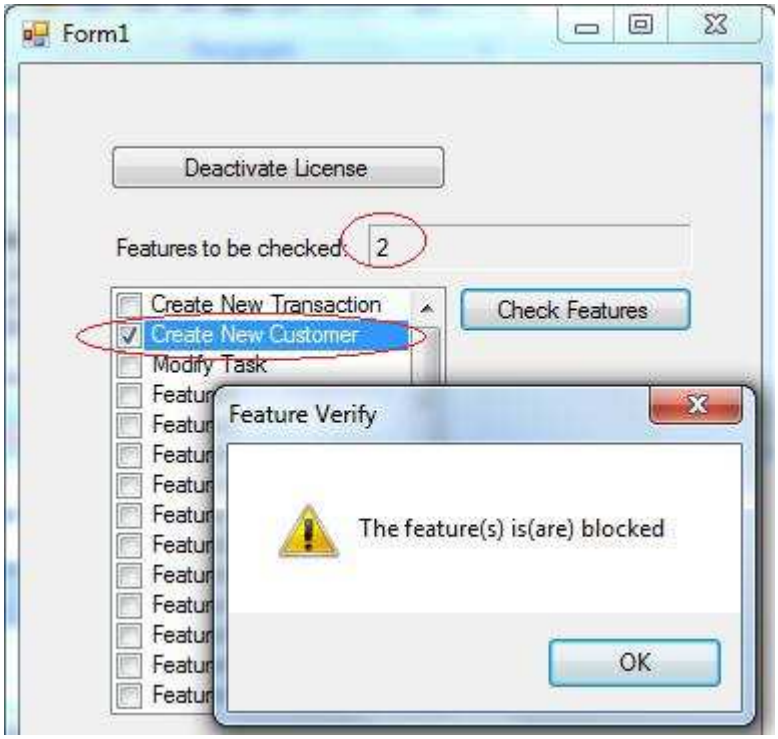


Demonstrations of verifying protected features

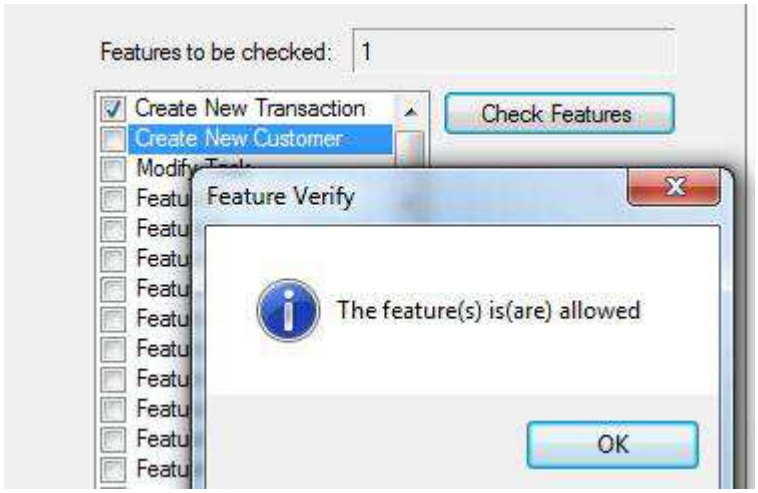
Your user gets the license file. Your program runs normally:

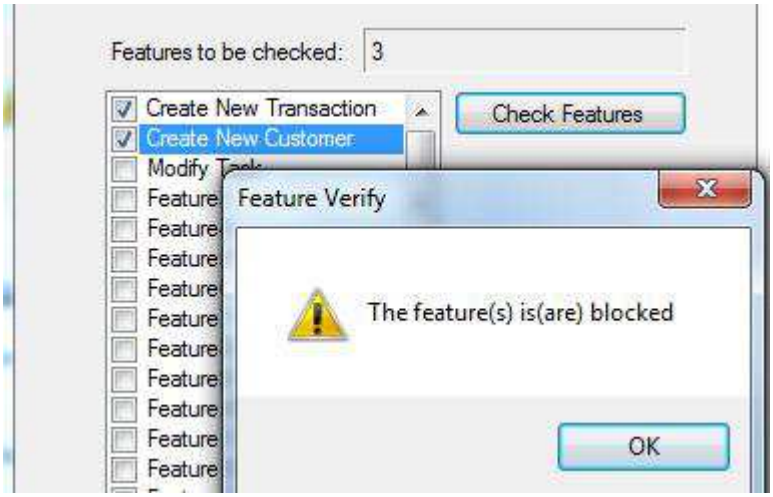


Select the second feature and click “Check Features”. We get “feature block” message. Note the integer value for selected features.

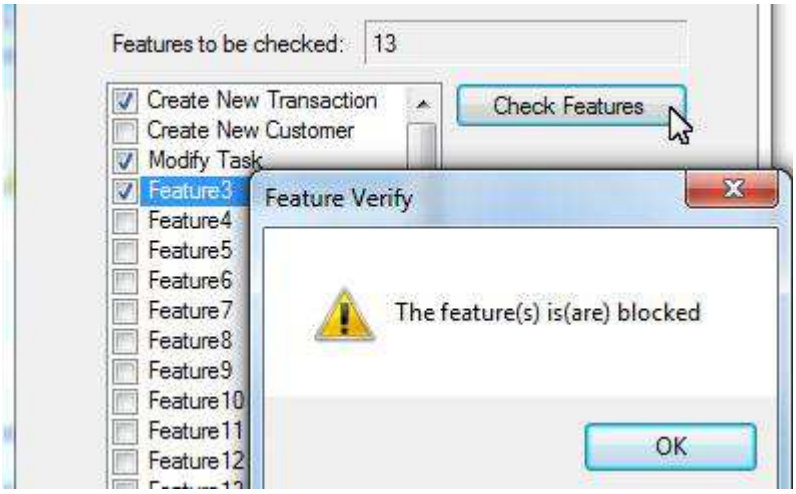


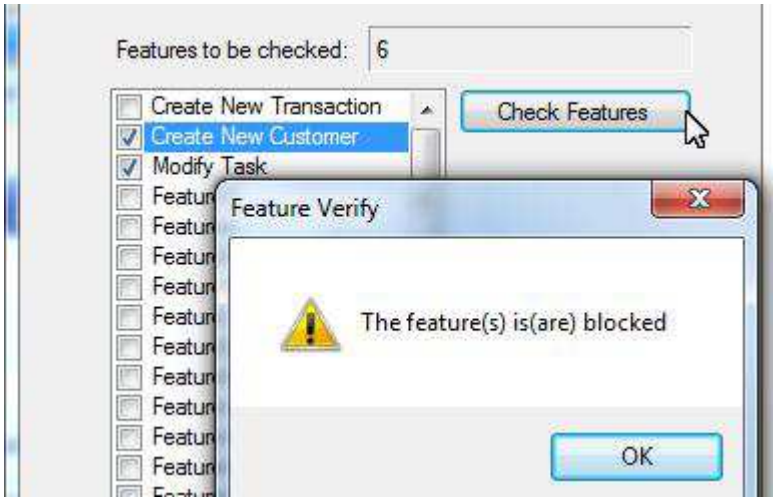
Below are more testing results:

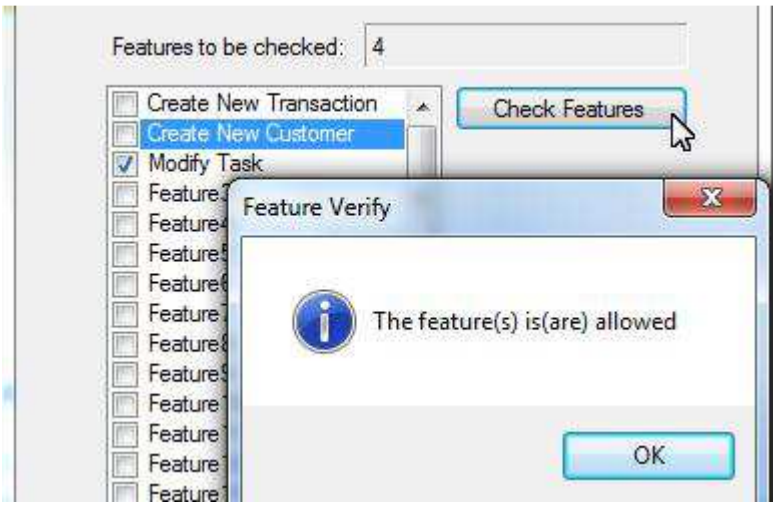



Features to be checked: 3
Create New Transaction
Create New Customer
Check Features
Feature Verify
The feature(s) is(are) blocked
OK


Features to be checked: 5
Create New Transaction
Create New Customer
Modify Task
Check Features
Feature Verify
The feature(s) is(are) allowed
OK


Features to be checked: 13
Create New Transaction
Create New Customer
Modify Task
Feature3
Feature4
Feature5
Feature6
Feature7
Feature8
Feature9
Feature10
Feature11
Feature12
Check Features
Feature Verify
The feature(s) is(are) blocked
OK





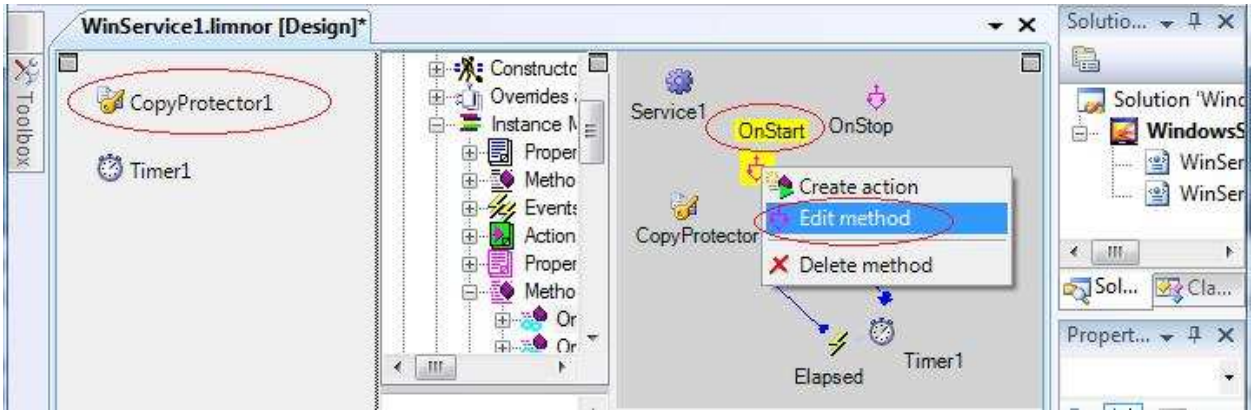
Activation of Services and Servers

Add CopyProtector to Service

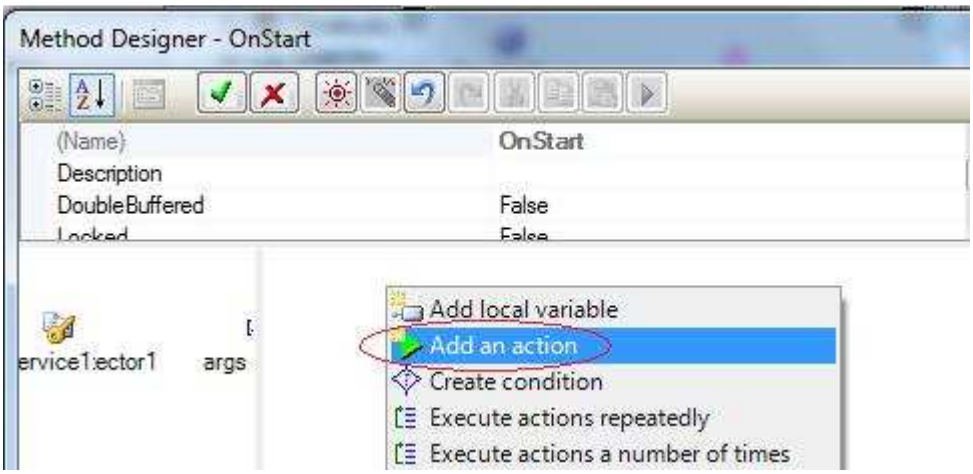
For Windows Service software, it does not use user interface. It cannot display the activation dialogue box. A separate utility program, AppLicense.exe, is provided to generate activation files.

We add a CopyProtector to a Windows Service, create a VerifyLicense action. We may add this action to the OnStart method so that the action will be executed when the service is being started.

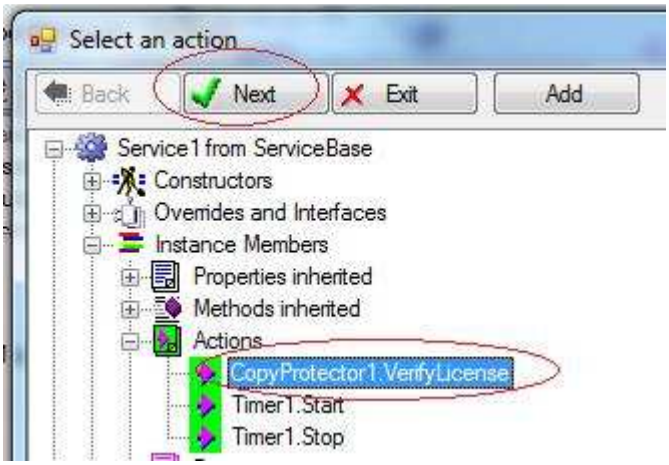
Right-click OnStart method, choose “Edit method”:



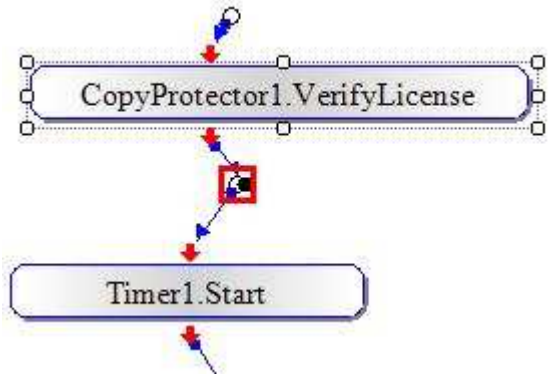
Right-click on the middle pane, choose “Add action”:



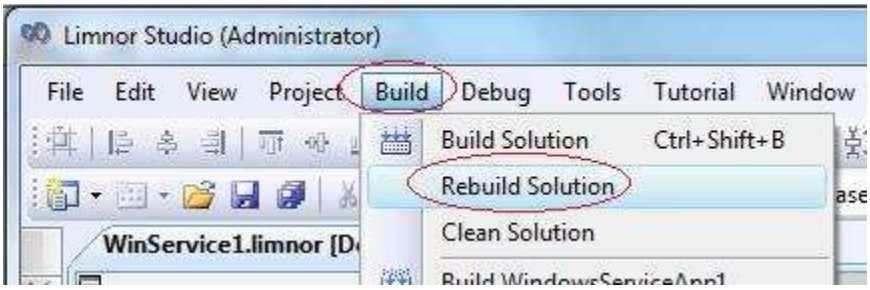
Select the VerifyLicense action, click Next:



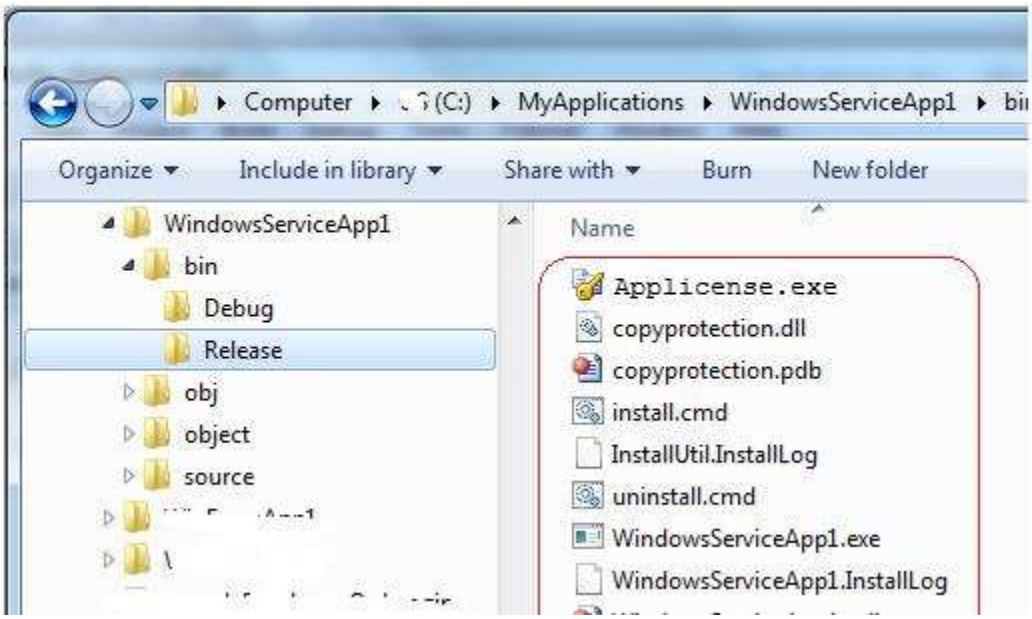
Link the VerifyLicense action to the first existing action to make the VerifyLicense the first action:



Compile this Windows Service project:



Compilation results are saved in bin\Release folder:

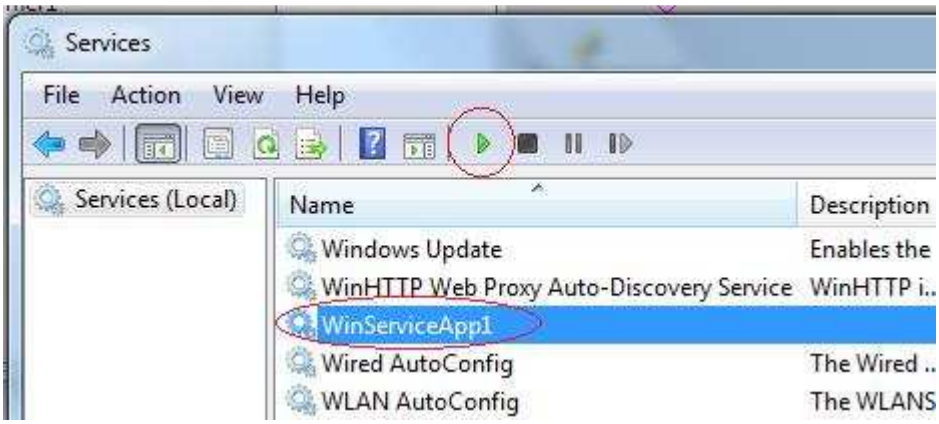


Distribution of the software is to copy the files to the user's computer. Note the program AppLicense.exe among the files. It is the utility for generating activation file.

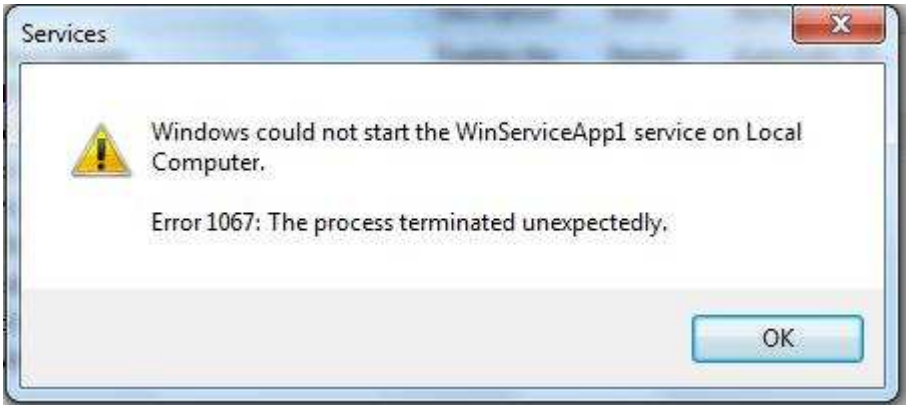
Suppose the files are distributed to a customer's computer. Run install.cmd to install the service:

Activate License

After installation, start the service:



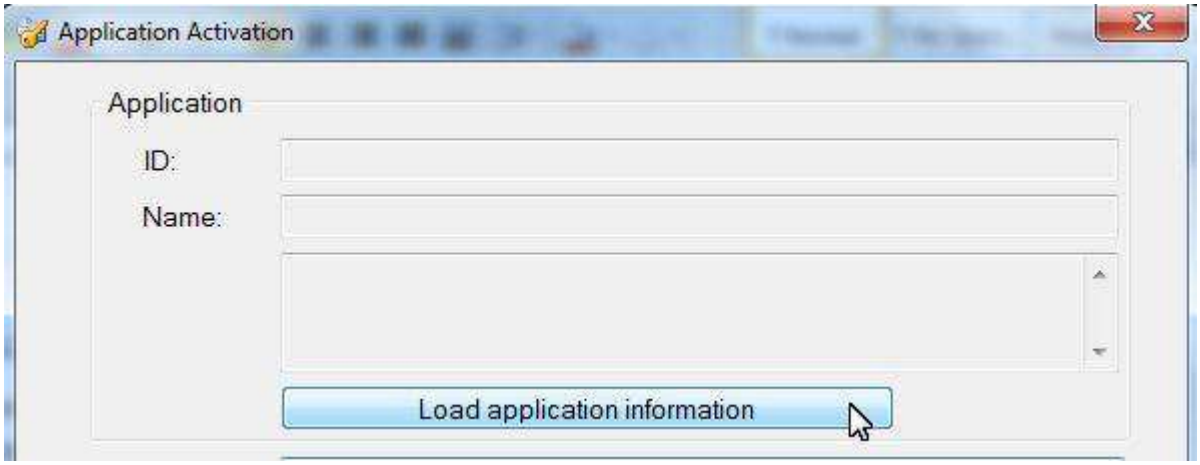
The starting of the service will fail:



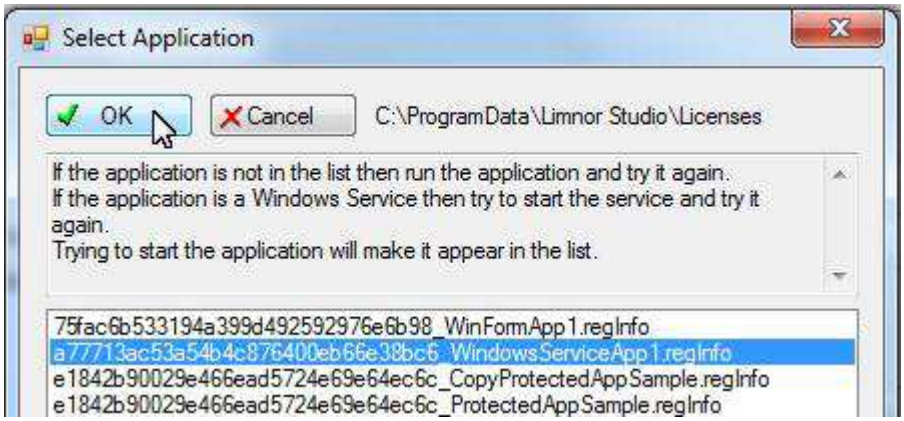
It failed because the license has not being issued. The user needs to generate an activation file using AppLicense.exe.

Note that it is important to let the service fail once because it will generate information for AppLicense.exe to use.

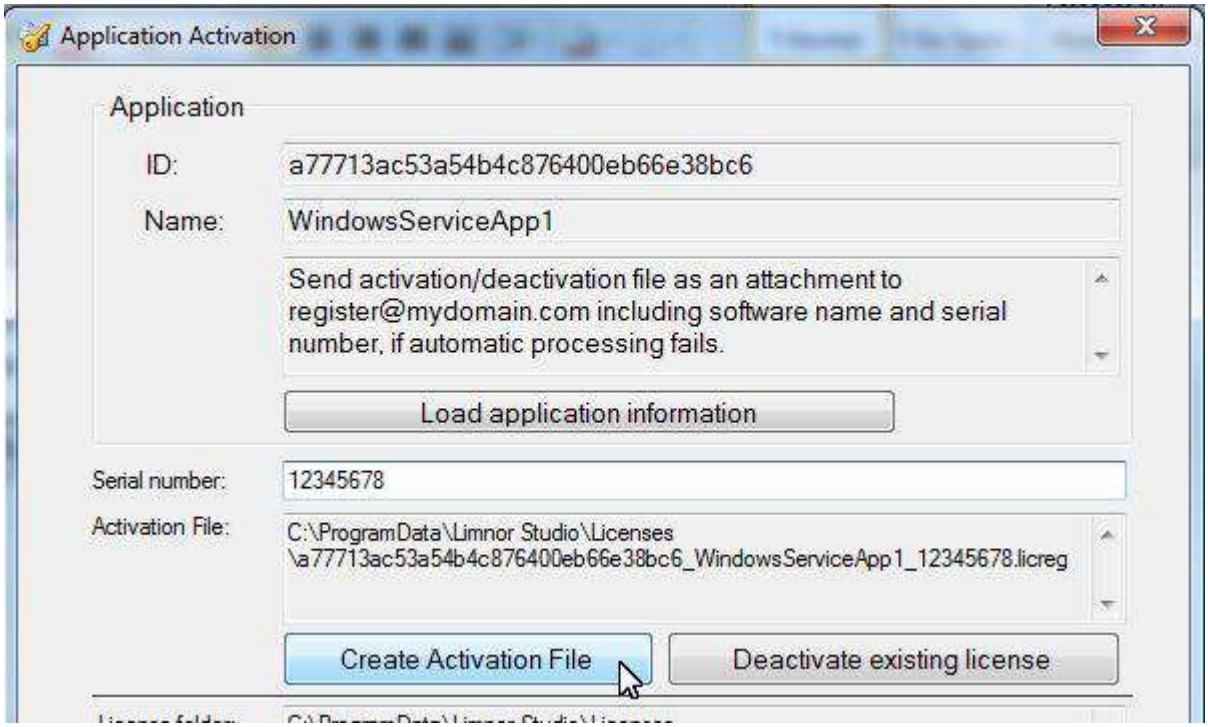
Run AppLicense.exe. Click button “Load application information”:



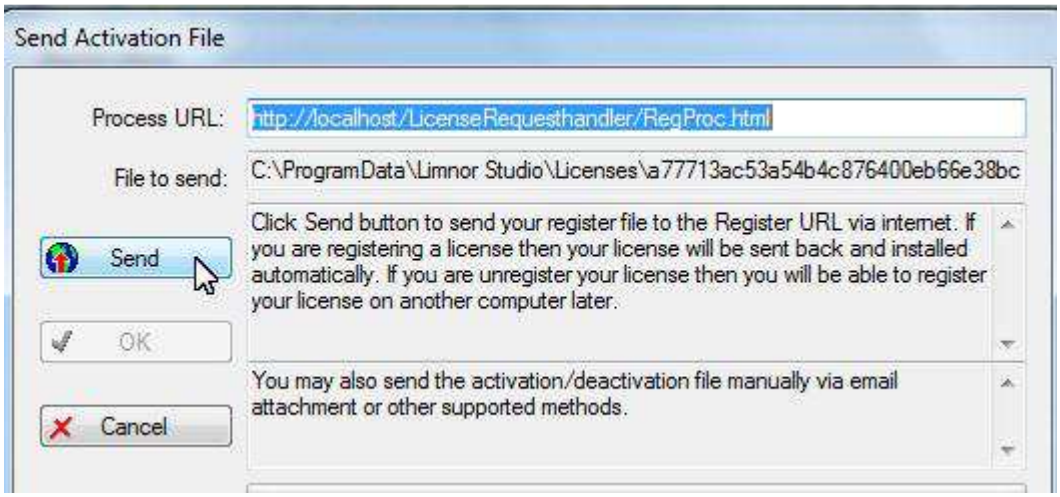
Select the service application, click OK:

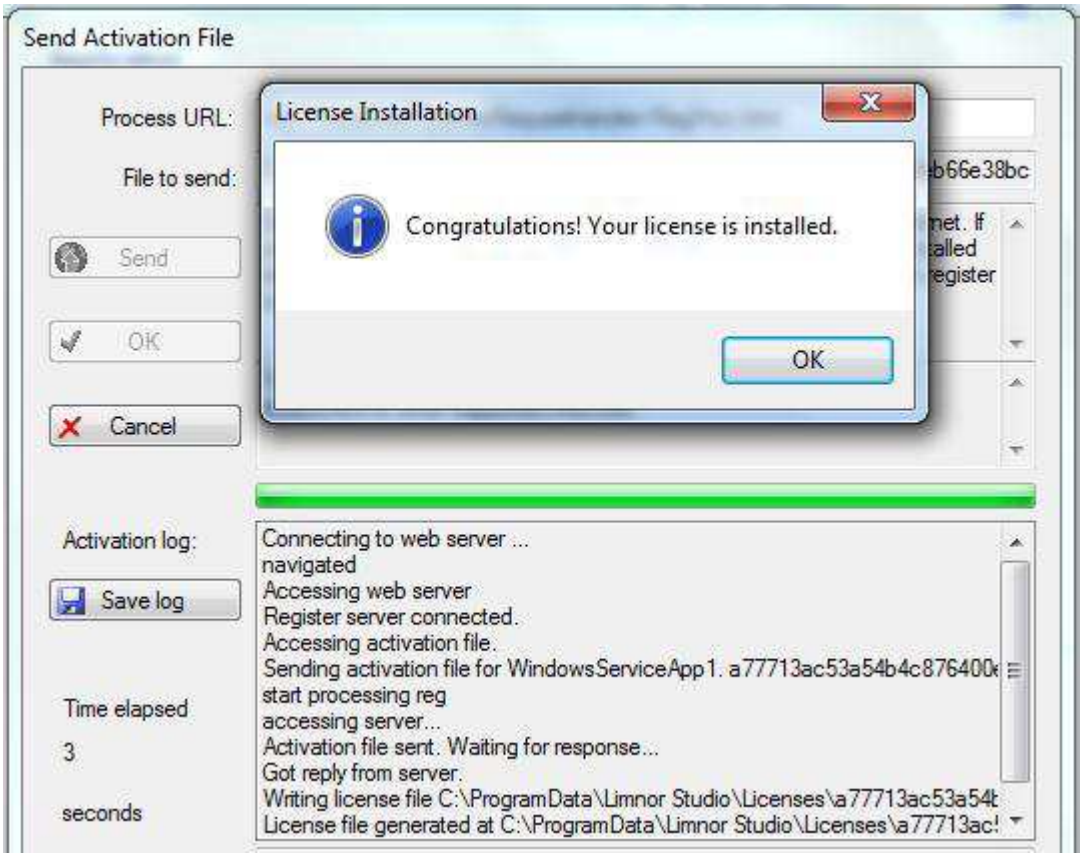


The information for the service appears in the dialogue. Enter the serial number. Click button “Create Activation File”:

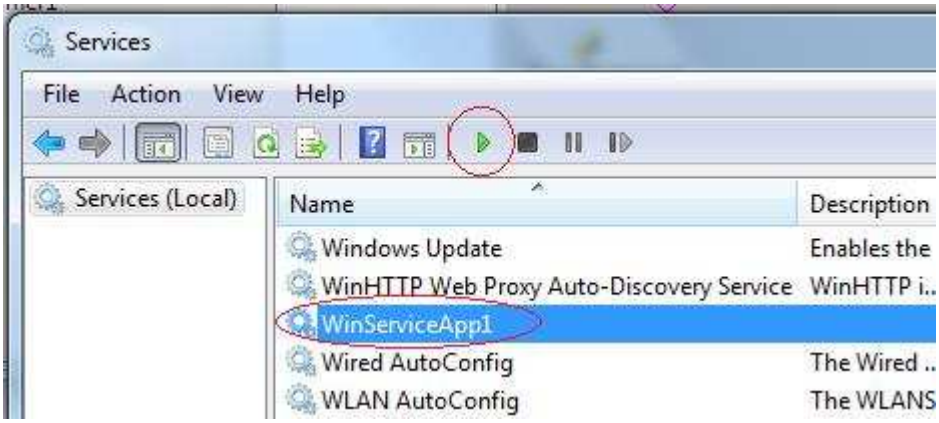


The activation file is created. A dialogue box appears for sending the activation file to your web server for automated processing.

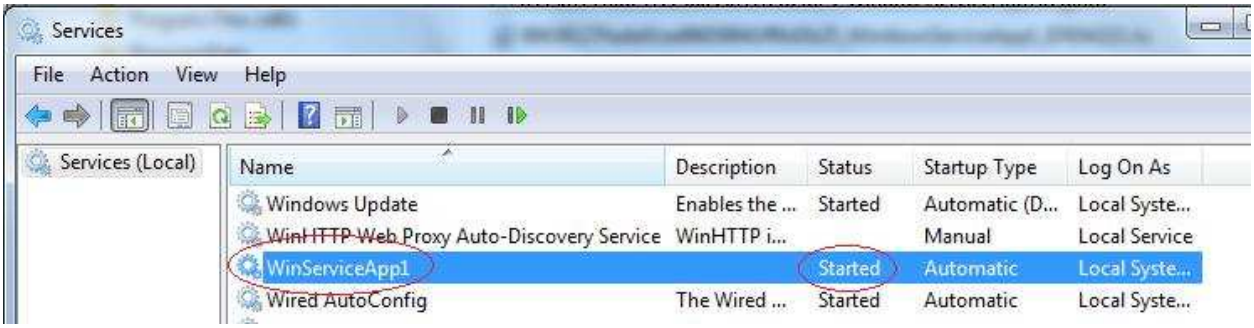




Now you may start the service:



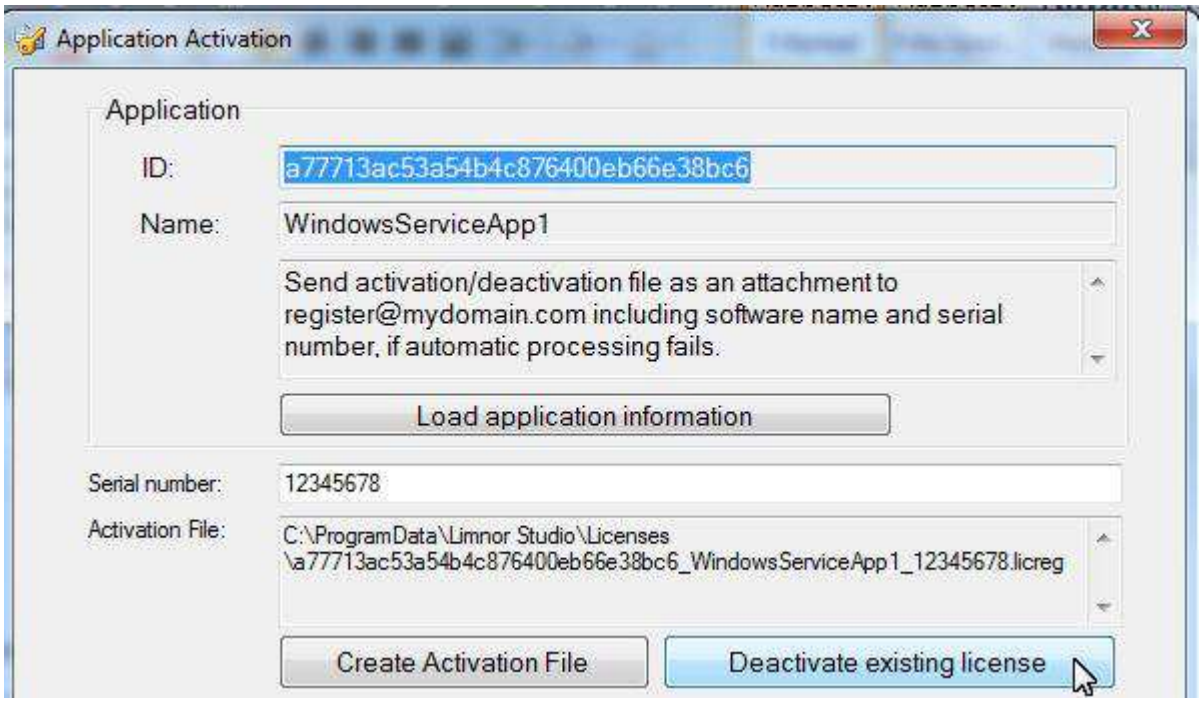
This time the service started successfully:



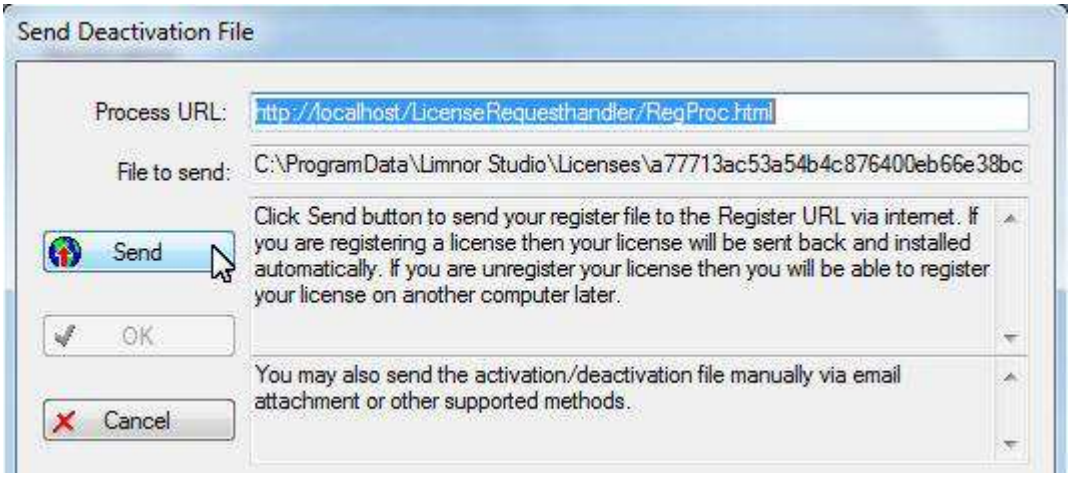
Deactivate License

AppLicense.exe may also be used to deactivate a license.

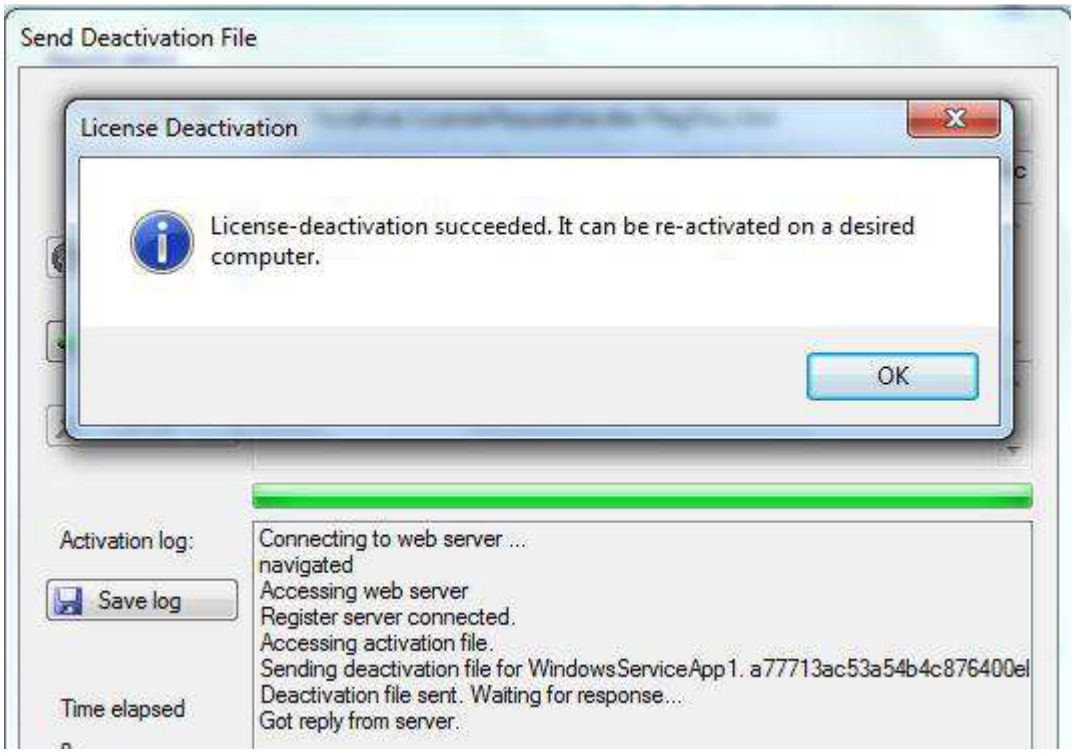
Load information for an application, enter serial number, then click “Deactivate existing license”:



A dialogue box appears to let the user send the deactivate message to your license management system:

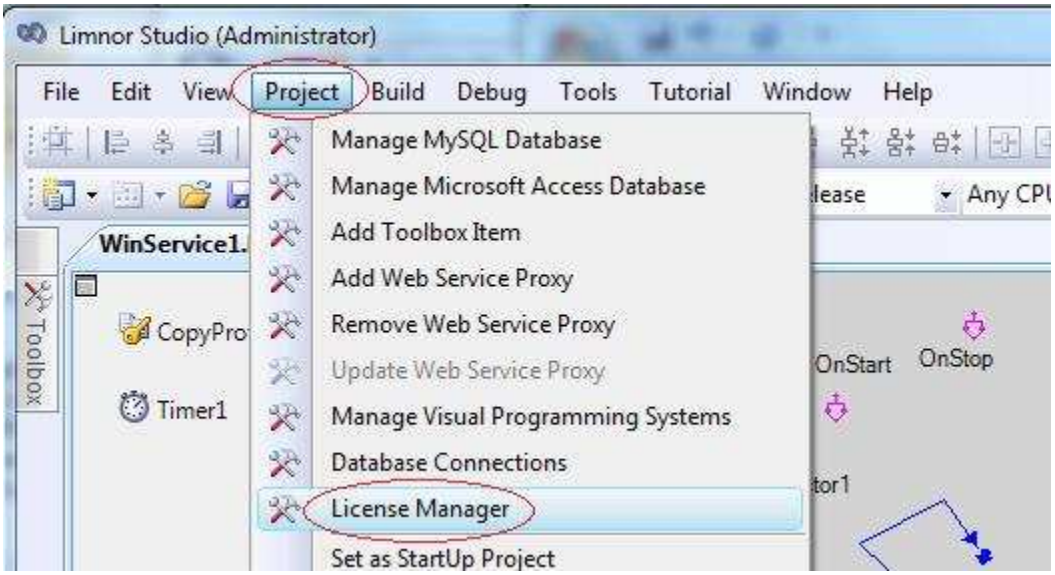


Click “Send”. The deactivation information is recorded in your license management system.



Manually Process License Activation

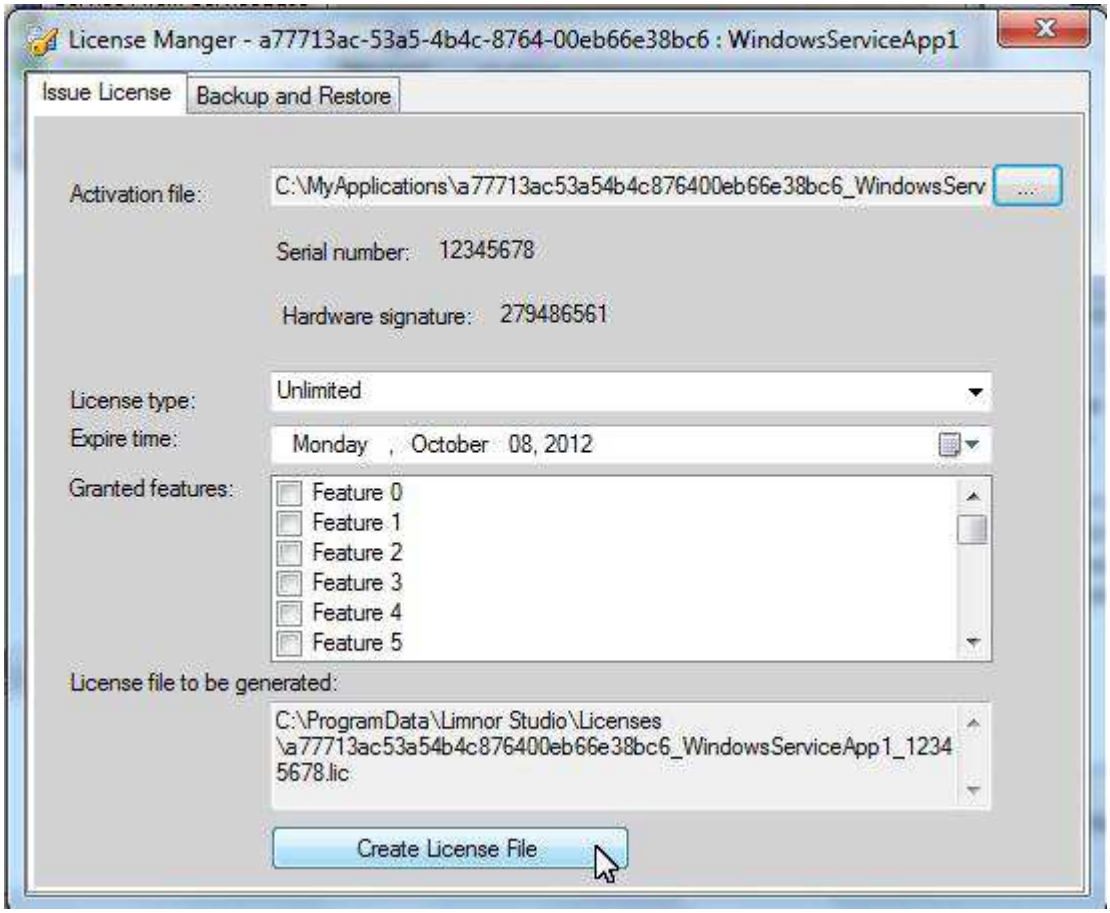
The user may also send the activation file to you. You open the Limnor Studio and load the service project. Open the “License Manager”:



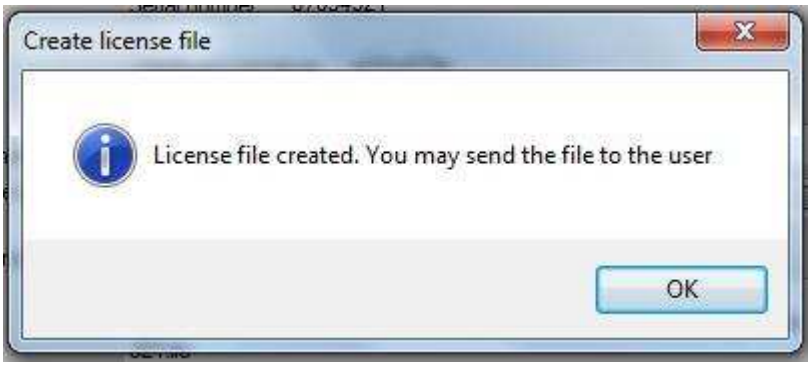
Click button  to load the activation file:

Select the activation file the customer submitted.

The information for the activation file appears. Click button “Create License File”:

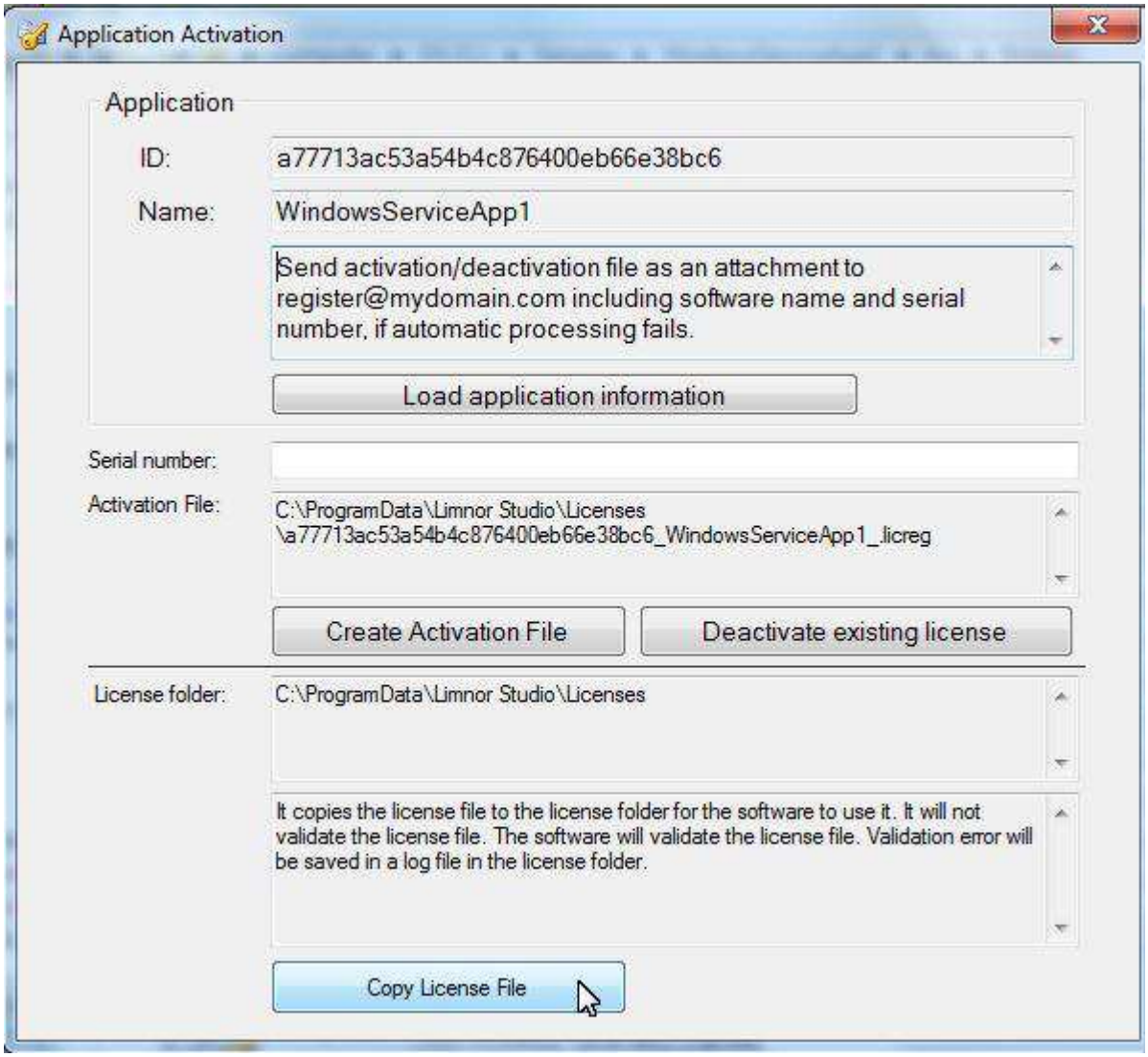


The license file is generated:

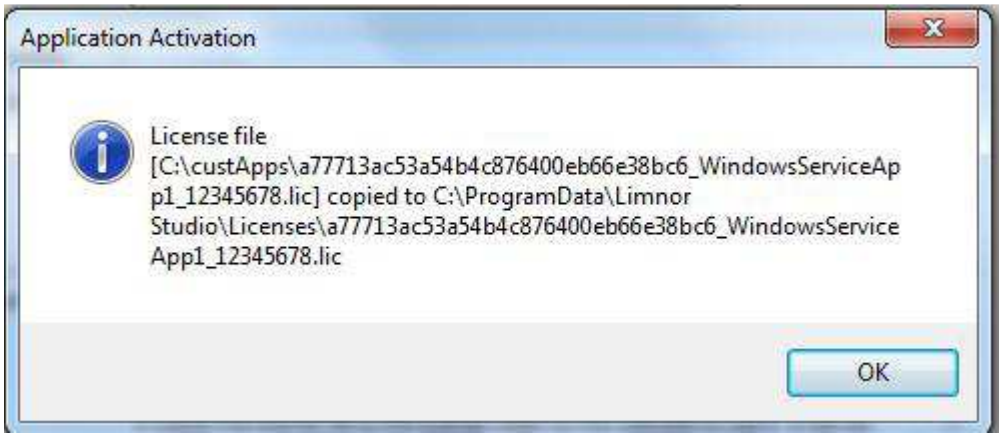


Send the license file to the user.

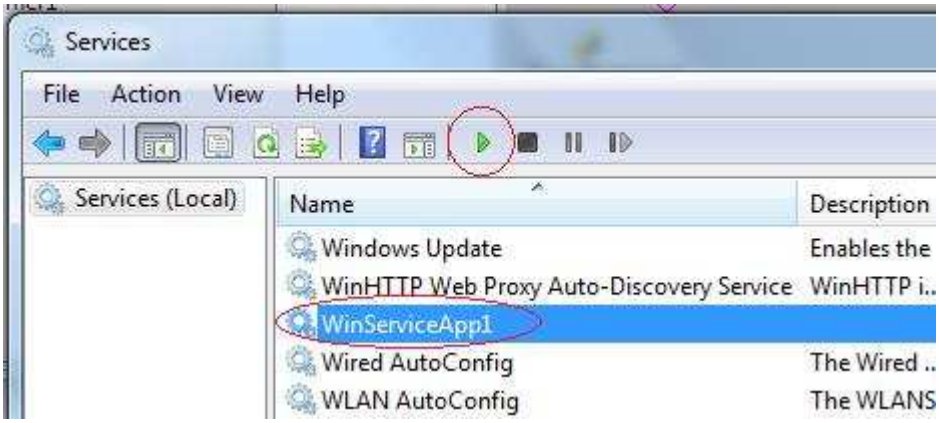
The user may also use AppLicense.exe utility to install the license file. Click “Copy License File”:



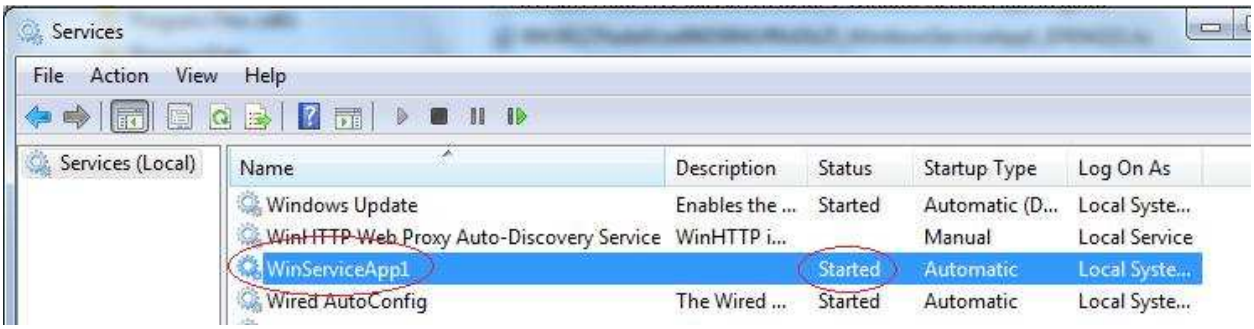
Select the license file from disk. The license file is copied to the license folder:



Now let's start the service again:



This time the service started successfully:

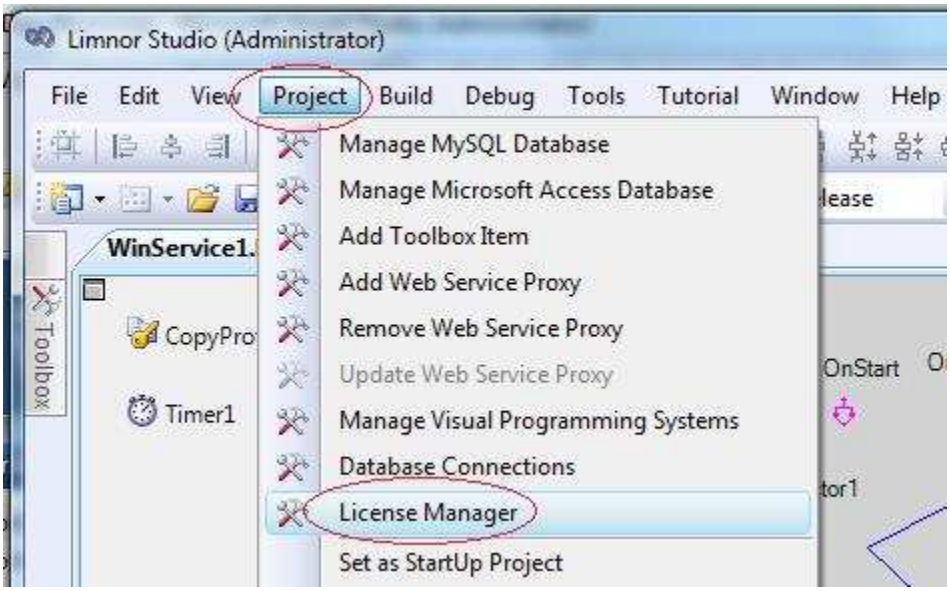


Backup and Restore License Manager

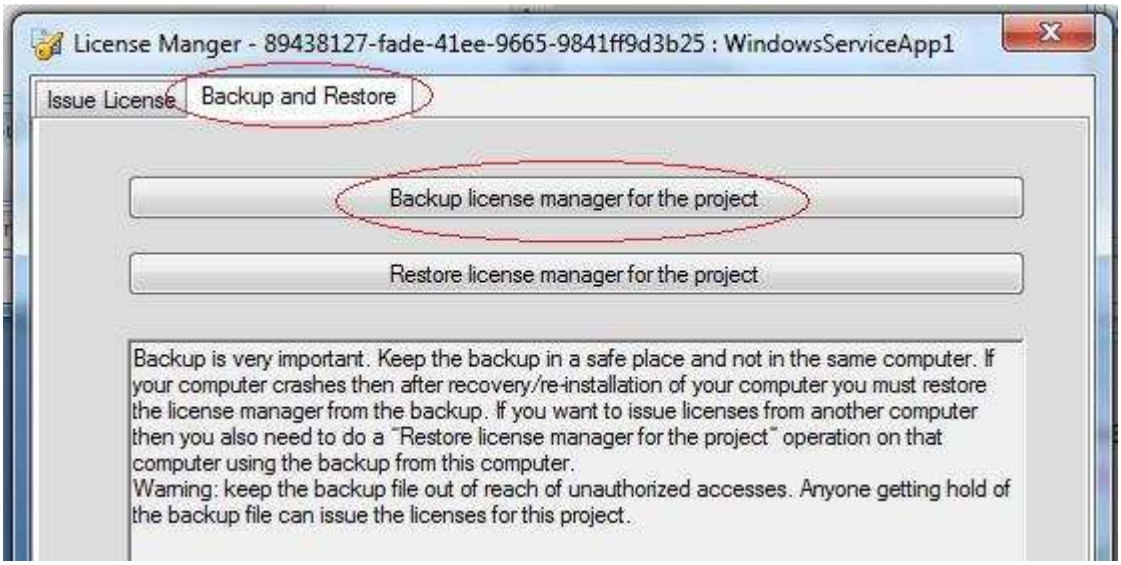
Initially the issuing of the licenses must be done from the computer compiling the software project. Suppose the computer crashes. The operating system is re-installed. Then it can no longer issue correct licenses for the project.

To ensure that correct licenses can always be issued, it is important to backup the License Manager.

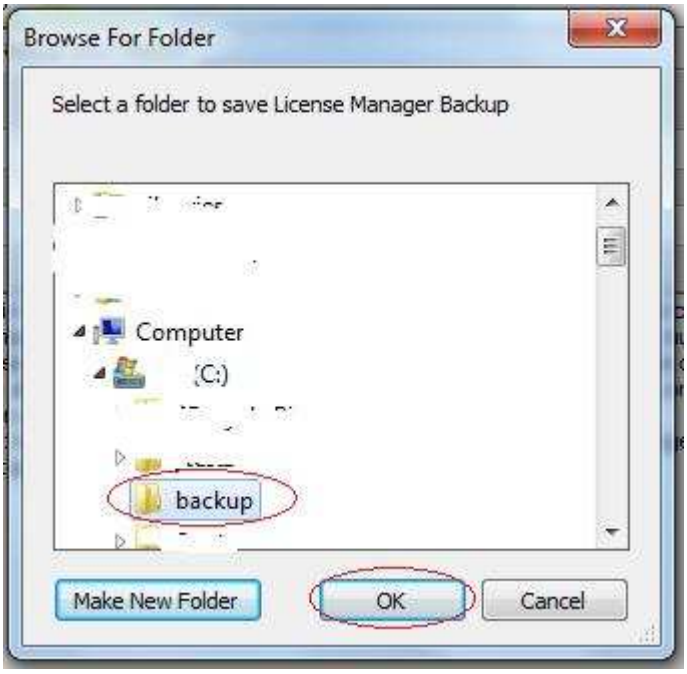
To do backup, open the License Manager:



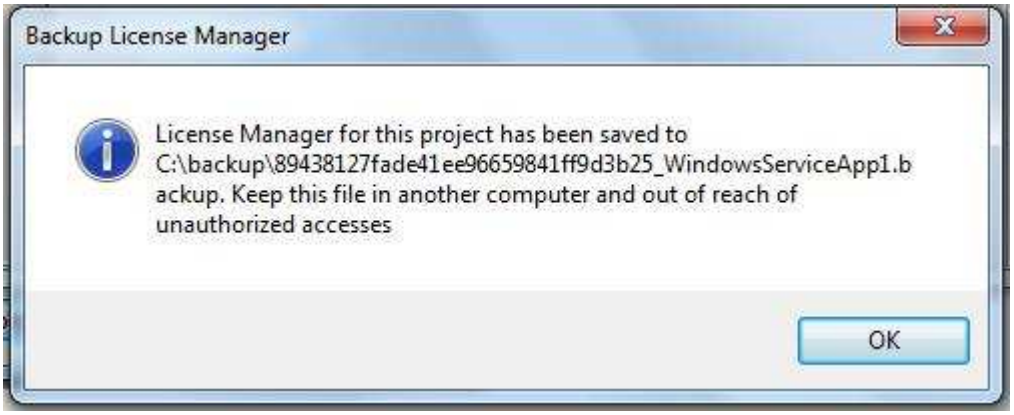
Choose “Backup and Restore”. Click button “Backup license manager for the project”:



Select a folder to save the backup file:



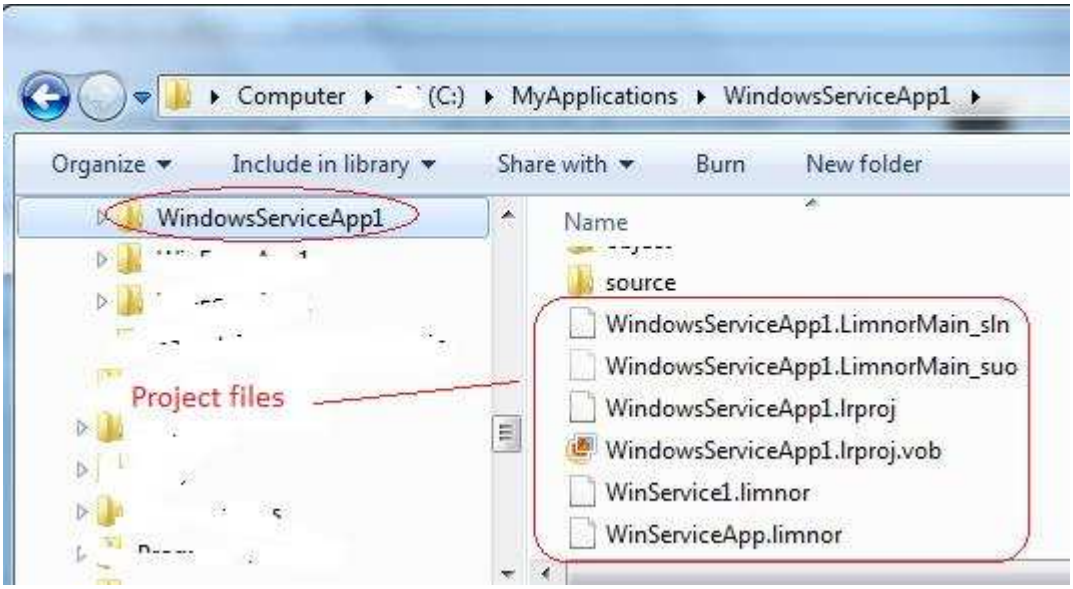
The backup file is saved to the folder selected:



The backup file contains all the information for issuing licenses for the project. You need to keep the backup file in a safe place. Do not allow unauthorized persons to get hold of it. Do not keep it only in the same computer used to issue licenses. If the computer crashes then the backup is also destroyed.

The backup file also allows you to issue licenses from any computers.

The project files should also be backed up:

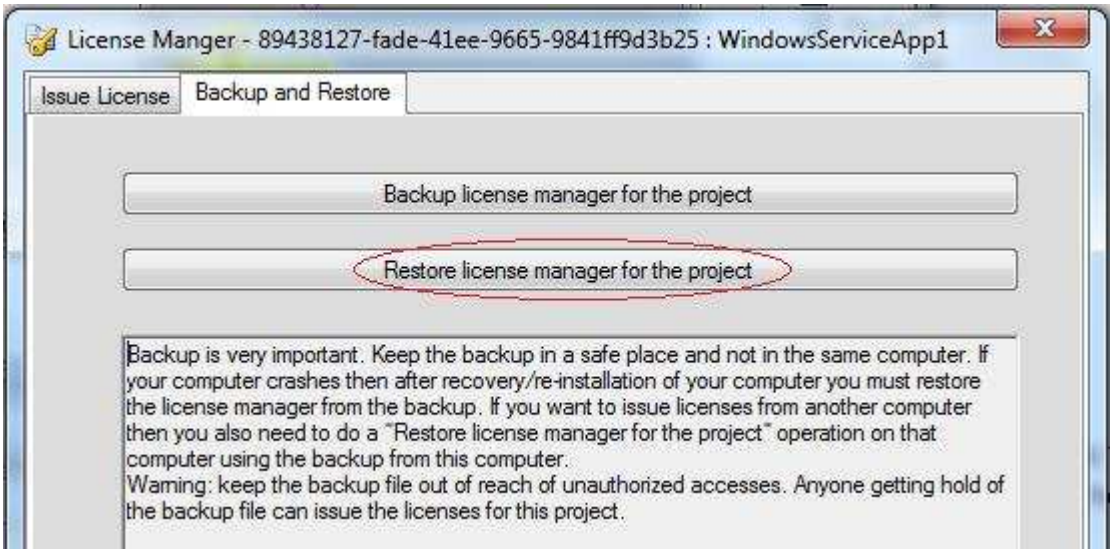


Suppose the computer crashed. The operating system is re-installed. Limnor Studio is re-installed. The project files are restored. We may use the backup file to enable the capability of issuing licenses for the project from the computer.

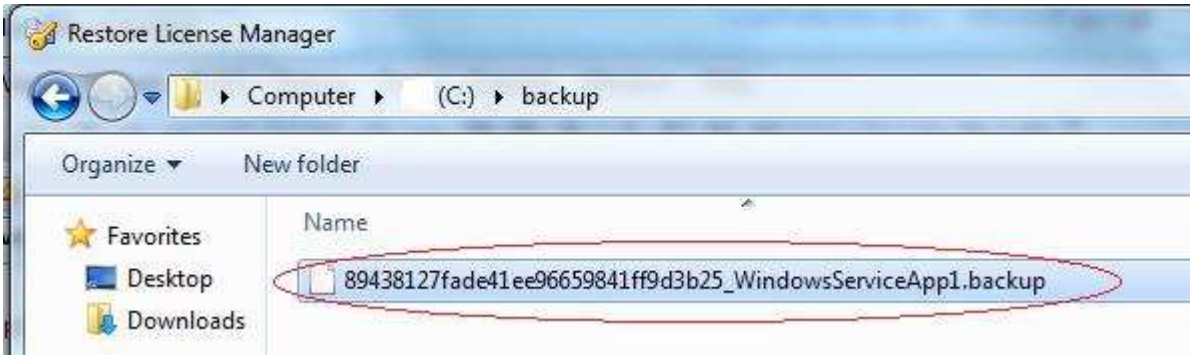
Or suppose the Limnor Studio is installed to a new computer and the project files are copied to the new computer. We may use the backup file to enable the capability of issuing licenses for the project from this new computer.

Open Limnor Studio, load the project. Open “License Manager”.

Click button “Restore license manager for the project”:



Select the backup file:



The backup is restored:

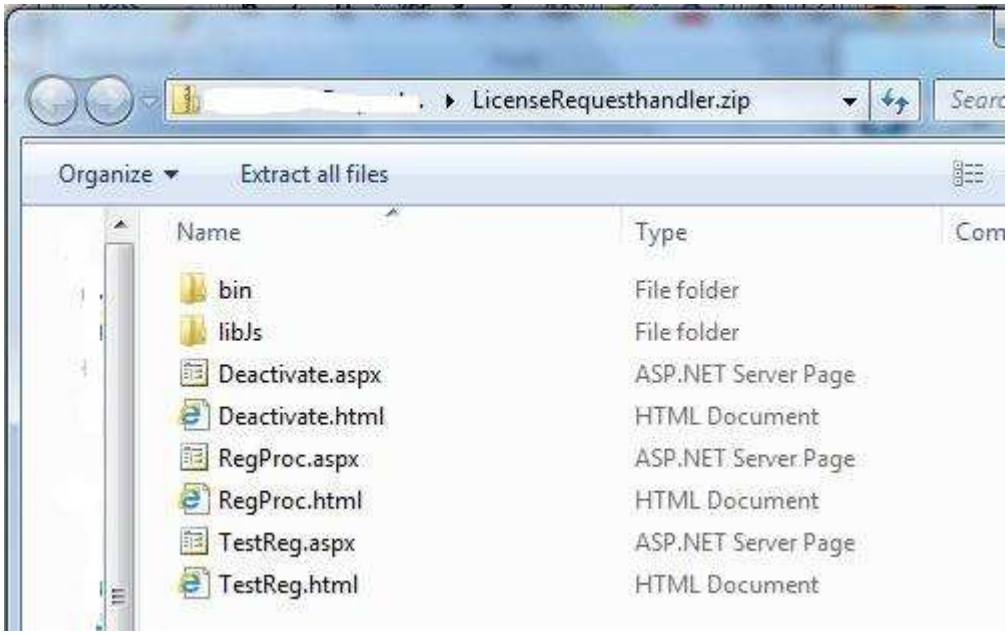
From now on, licenses for this project can be issued from this computer.

Automated License Activation/Deactivation

Get Web Files

You may download files for your web server from http://www.limnor.com/studio/licenseWeb.zip

The zip file contains HTML files and supporting files:



This is a web application created using Limnor Studio. You may use it as it is because it does not require specific parameters such as database connections. All your specific business logic will be in a separate license management plug-in. A plug-in sample, also created using Limnor Studio, can be found at http://www.limnor.com/studio/licensePlugin.zip

Deploy to Web Server

You need to have an IIS web server supporting Aspx.Net and supporting .Net Framework 3.5.

Suppose your web site is http://www.mydomain.com and you setup a virtual folder named LicenseRequesthandler. Then your web URL to be set in CopyProtector components is http://www.mydomain.com/LicenseRequesthandler/RegProc.html.

Suppose the physical folder for the above virtual folder is C:\inetpub\wwwroot\LicenseRequesthandler then you need to unzip **LicenseRequesthandler.zip** to C:\inetpub\wwwroot\LicenseRequesthandler and keep the folder structure of the zip files.

If your web server uses Windows XP then the bin folder should be under your web root folder.

Deploy License Manager

In the chapter of "Backup and Restore License Manager", we described how to create a license manager backup file. You need to copy the backup file to your web physical folder. For example, copy file **e1842b90029e466ead5724e69e64ec6c_CopyProtectedAppSample.backup** to folder C:\inetpub\wwwroot\LicenseRequesthandler.

Create a Licenses folder

The web application needs a folder to save the license files it will generate. The folder is named "Licenses". If your web physical folder is C:\inetpub\wwwroot\LicenseRequesthandler then you need to

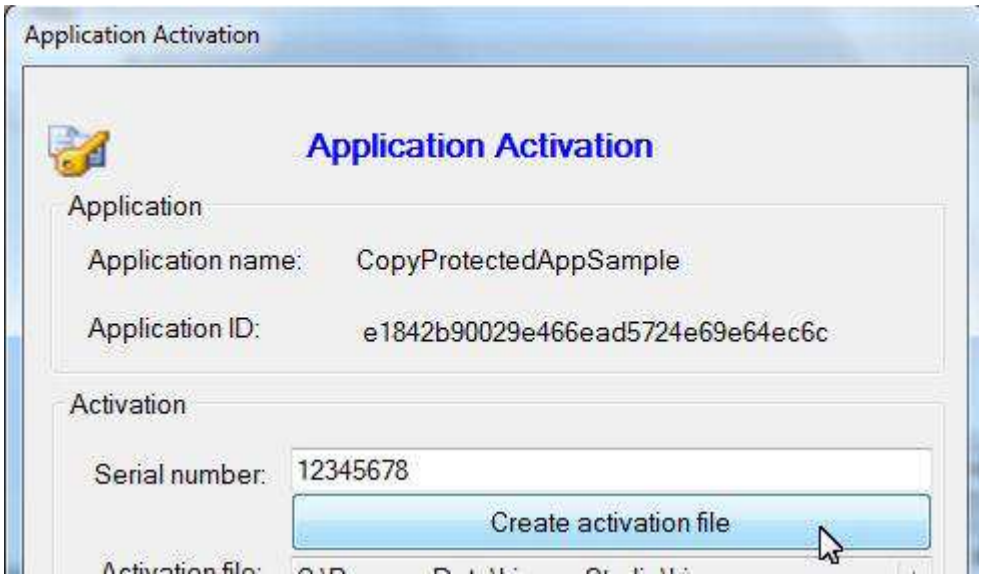
create a folder C:\inetpub\wwwroot\LicenseRequesthandler\Licenses. Make sure that your web application has the write-permission for the Licenses folder.

Test Deployment

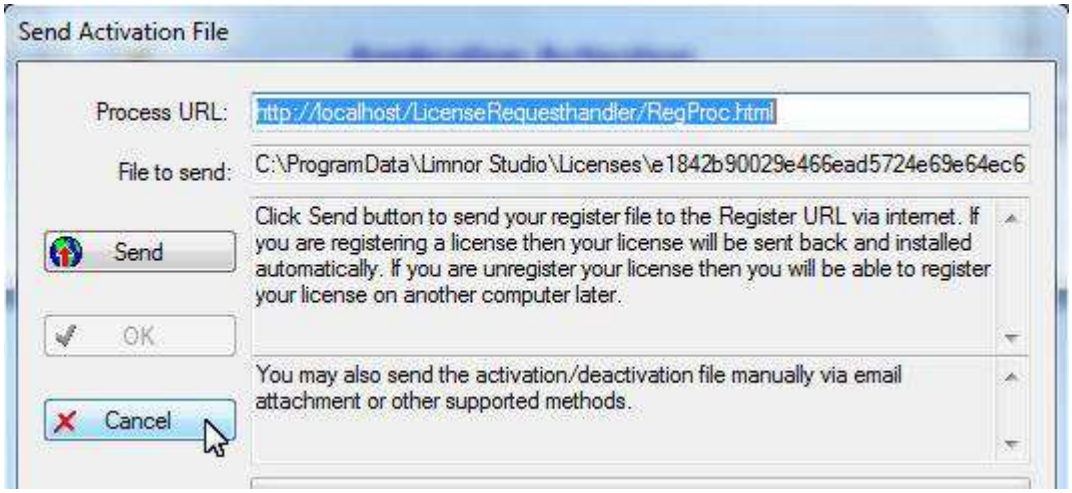
Your processing URL, http://www.mydomain.com/LicenseRequesthandler/RegProc.html, cannot run directly from a web browser. There is a test web page, testReg.html, for you to test your web file deployment. You may launch http://www.mydomain.com/LicenseRequesthandler/TestReg.html from a web browser.

Generate test information

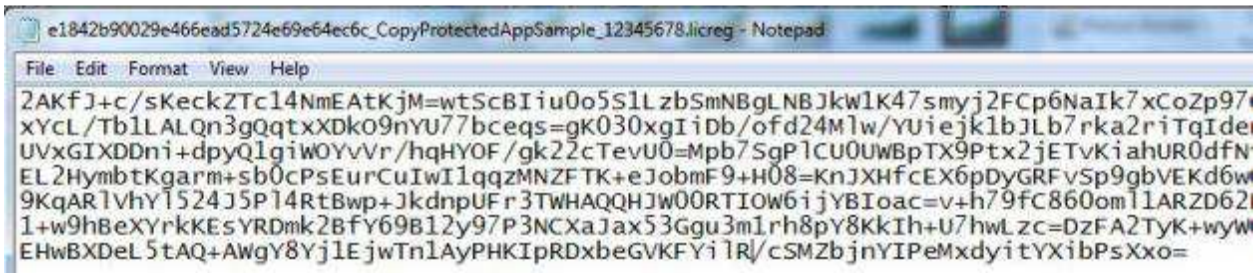
To make a test, first run your protected application to generate a license activation file.



Cancel the sending of the activation file.



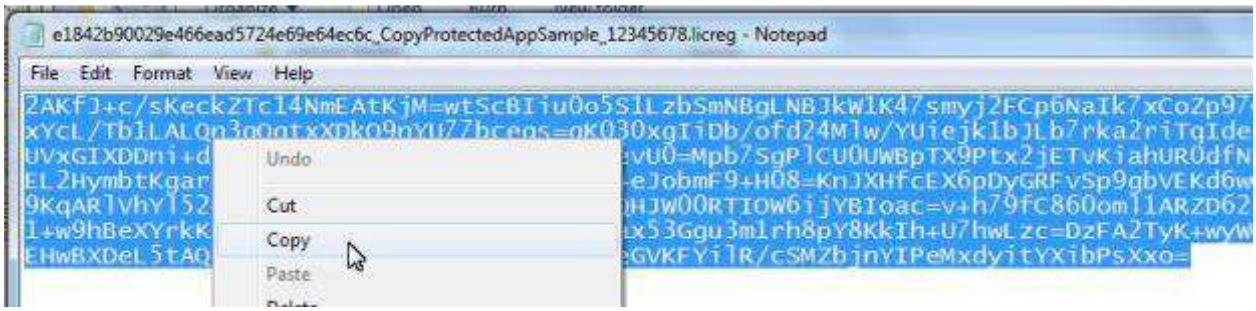
Use the Notepad or other text file reader to open the activation file:



Use test web page

Launch http://www.mydomain.com/LicenseRequesthandler/TestReg.html from a web browser.

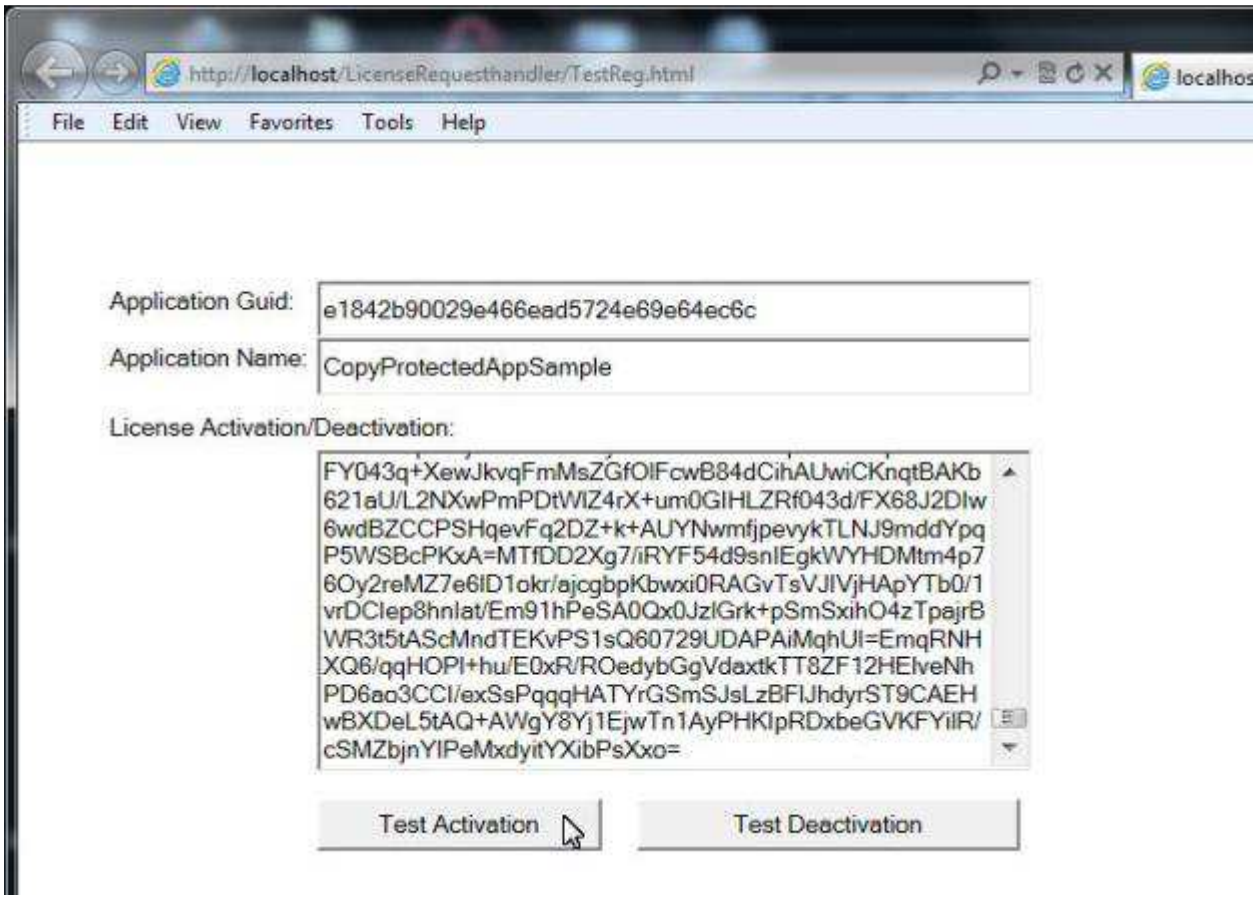
Copy whole contents of the activation file:



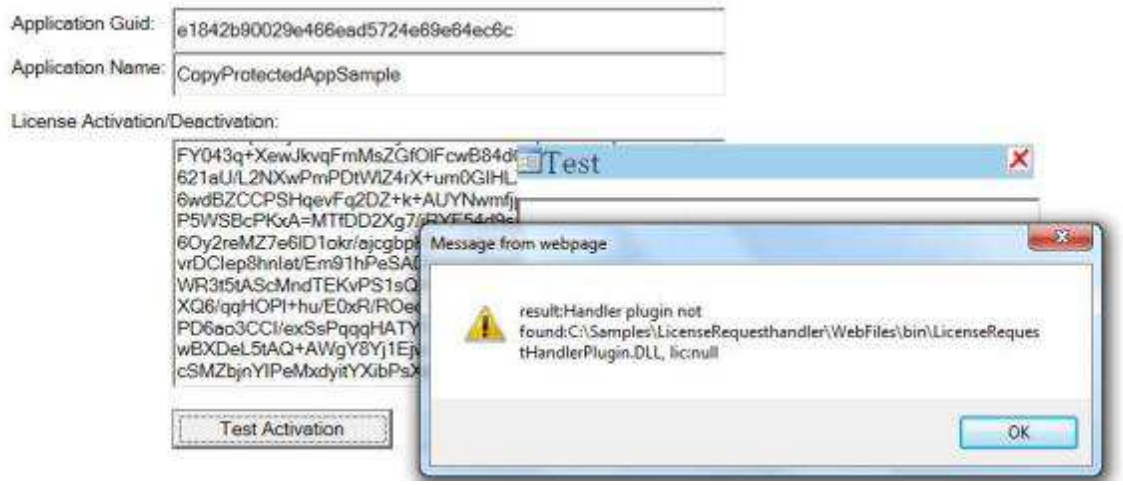
Paste it to the text box on the test web page:

Enter application GUID and application name for your application.

Click “Test Activation” on the test web page.



At this stage you will get an error message telling you that it cannot find license management plug-in.



If you reached this error message then you have successfully deployed your web processing server.

Now you need to create your license management plug-in. The plug-in is responsible for issuing licenses and recording license activation and deactivation.

We are going to show a sample license management plug-in.

License Management Plug-in Sample

This sample uses a MySQL database to manage license activation and deactivation. You may certainly use any other databases as long as accessing from .Net Framework is supported. This sample can be downloaded from http://www.limnor.com/studio/licensePlugin.zip.

Database schema

Below is a MySQL script for creating MySQL database objects used in this plug-in sample.

```
/*Begin of installation*/
/*programs to be protected*/
CREATE TABLE softwareProtected (
     SoftwareID INT NOT NULL AUTO_INCREMENT,
     CreateTime DATETIME NULL,
     AppGuid VARCHAR(40) NOT NULL,
     AppName VARCHAR(80) NOT NULL,
     PRIMARY KEY (SoftwareID)
);
/*license activation and deactivation*/
CREATE TABLE softwareRegister (
     RegisterID INT NOT NULL AUTO_INCREMENT,
     SoftwareID INT NOT NULL,
     RegisterTime DATETIME NULL,
     SerialNumber VARCHAR(60) NOT NULL,
     HardwareSignature INT NULL,
     RegisterKey VARCHAR(33) NULL,
     LicenseCode INT NULL,
     Features INT NULL,
     ExpireTime DATETIME NULL,
     RegisterDescID INT NULL,
     SoftwareManagerID INT NULL,
     PRIMARY KEY (RegisterID)
);
/*notes for license activation and deactivation*/
CREATE TABLE softwareRegisterDesc (
     RegisterDescID INT NOT NULL AUTO_INCREMENT,
```

```
     RegisterType INT NULL,
     RegisterDescription TEXT NULL,
     PRIMARY KEY (RegisterDescID)
);
/*serial numbers issued*/
CREATE TABLE softwareLicense (
     SoftwareID INT NOT NULL,
     SerialNumber VARCHAR(60) NOT NULL,
     CreateTime DATETIME NULL,
     RegisterID INT NULL,
     InstallingFeatures INT NULL,
     InstallingLicenseCode INT NULL,
     InstallingExpireTime DATETIME NULL,
     PRIMARY KEY (SoftwareID,SerialNumber)
);
/*persons for manually license activation/deactivation*/
CREATE TABLE softwareManager (
     SoftwareManagerID INT NOT NULL AUTO_INCREMENT,
     ManagerAlias VARCHAR(50) NOT NULL,
     ManagerPass VARCHAR(50) NOT NULL,
     ManagerName VARCHAR(50) NULL,
     ManagerLevel int NULL,
     PRIMARY KEY (SoftwareManagerID)
);

CREATE UNIQUE INDEX softwareManagerAlias ON softwareManager (ManagerAlias);
CREATE UNIQUE INDEX softwareProtectedGuid ON softwareProtected (AppGuid);
CREATE INDEX softwareRegisterSN ON softwareRegister (SerialNumber);
CREATE UNIQUE INDEX softwareRegisterKey ON softwareRegister (RegisterKey);
ALTER TABLE softwareLicense ADD
      CONSTRAINT FK_softwareLicense_softwareProtected
      FOREIGN    KEY (SoftwareID)
      REFERENCES softwareProtected(SoftwareID);
ALTER TABLE softwareLicense ADD
      CONSTRAINT FK_softwareLicense_softwareRegister
      FOREIGN    KEY (RegisterID)
      REFERENCES softwareRegister(RegisterID);
ALTER TABLE softwareRegister ADD
      CONSTRAINT FK_softwareRegister_softwareProtected
      FOREIGN    KEY (SoftwareID)
      REFERENCES softwareProtected(SoftwareID);
ALTER TABLE softwareRegister ADD
      CONSTRAINT FK_softwareRegister_softwareRegisterDesc
      FOREIGN    KEY (RegisterDescID)
      REFERENCES softwareRegisterDesc(RegisterDescID);
ALTER TABLE softwareRegister ADD
      CONSTRAINT FK_softwareRegister_softwareManager
      FOREIGN    KEY (SoftwareManagerID)
      REFERENCES softwareManager(SoftwareManagerID);

/*automatic license activation*/
DELIMITER //
CREATE PROCEDURE ActivateLicense (IN appGuid VARCHAR(40), IN sn VARCHAR(60), IN regKey
VARCHAR(33), IN hardwareSignature INT, OUT ret VARCHAR(500))
BEGIN
       DECLARE softId INT;
       DECLARE userLicCode INT;
       DECLARE userFeatures INT;
       DECLARE userExpTime DATETIME;
       DECLARE licCode INT;
       SET ret = 'Fail;Unknown error';
       IF Exists(SELECT 1 FROM softwareRegister WHERE RegisterKey=regKey) THEN
              SET ret = CONCAT('Fail;registry already used:',regKey);
       ELSE
              SELECT
                     softwareLicense.SoftwareID,
                     softwareLicense.InstallingLicenseCode,
                     softwareLicense.InstallingFeatures,
                     softwareLicense.InstallingExpireTime,
                     softwareRegister.LicenseCode
              INTO softId,userLicCode,userFeatures,userExpTime,licCode
```

```
                FROM (softwareLicense INNER JOIN softwareProtected
                        ON softwareLicense.SoftwareID=softwareProtected.SoftwareID)
                LEFT JOIN softwareRegister
                        ON softwareLicense.RegisterID = softwareRegister.RegisterID
                WHERE softwareprotected.AppGuid = appGuid AND softwarelicense.SerialNumber = sn;
                IF softId IS NULL THEN
                        SET ret = CONCAT('Fail;invalid serial number:',sn);
                ELSE
                        IF licCode > 0 THEN
                                SET ret = 'Fail;license already activated';
                        ELSE
                                IF userLicCode IS NULL OR userLicCode = 0 THEN
                                        SET ret = CONCAT('Fail;the serial number is
inactivae:',CAST(IFNULL(userLicCode, 'NULL') AS CHAR));
                                ELSE
                                        INSERT softwareRegister (
                                                SoftwareID,
                                                RegisterTime,
                                                SerialNumber,
                                                HardwareSignature,
                                                RegisterKey,
                                                LicenseCode,
                                                Features,
                                                ExpireTime
                                        ) VALUES (
                                                softId,
                                                (SELECT NOW()),
                                                sn,
                                                hardwareSignature,
                                                regKey,
                                                userLicCode,
                                                userFeatures,
                                                userExpTime
                                        );
                                        UPDATE softwareLicense SET RegisterID=(SELECT
LAST_INSERT_ID()) WHERE SoftwareID=softId AND SerialNumber=sn;
                                        SET ret = CONCAT('OK;',CAST(userLicCode AS CHAR), ';',
CAST(userFeatures AS CHAR), ';', IFNULL(DATE_FORMAT(userExpTime, '%Y-%m-%d'),''));
                                END IF;
                        END IF;
                END IF;
        END IF;
END//
DELIMITER ;

/*automatic license deactivation*/
DELIMITER //
CREATE PROCEDURE DeactivateLicense (IN appGuid VARCHAR(40), IN sn VARCHAR(60), IN regKey
VARCHAR(33), IN hardwareSignature INT, OUT ret VARCHAR(500))
BEGIN
        DECLARE softId INT;
        DECLARE licCode INT;
        DECLARE currentRegKey VARCHAR(33);
        DECLARE fs INT;
        DECLARE expTime DATETIME;
        SET ret = 'Unknown error';
        SELECT
                softwareLicense.SoftwareID,
                softwareRegister.LicenseCode,
                softwareRegister.RegisterKey,
                softwareRegister.Features,
                softwareRegister.ExpireTime
        INTO softId, licCode, currentRegKey, fs, expTime
        FROM (softwareLicense INNER JOIN softwareProtected
                ON softwareLicense.SoftwareID=softwareProtected.SoftwareID)
    LEFT JOIN softwareRegister
                ON softwareLicense.RegisterID = softwareRegister.RegisterID
        WHERE softwareprotected.AppGuid = appGuid AND softwarelicense.SerialNumber = sn;
        IF softId IS NULL THEN
                SET ret = CONCAT('invalid serial number:', sn, ', app:', appGuid);
        ELSE
```

```
		IF licCode IS NULL OR licCode = 0 THEN
			SET ret = CONCAT('a license is not activated for serial number:', sn, ',
app:', appGuid);
		ELSE
			IF currentRegKey != regKey THEN
				SET ret = CONCAT('invalid register key [', regKey, '] for serial
number:', sn, ', app:', appGuid);
			ELSE
				INSERT softwareRegister (
					SoftwareID,
					RegisterTime,
					SerialNumber,
					HardwareSignature,
					RegisterKey,
					LicenseCode,
					Features,
					ExpireTime
				) VALUES (
					softId,
					(SELECT NOW()),
					sn,
					hardwareSignature,
					CONCAT('U', regKey),
					0,
					fs,
					expTime
				);
				UPDATE softwareLicense SET RegisterID=(SELECT LAST_INSERT_ID())
WHERE SoftwareID=softId AND SerialNumber=sn;
				SET ret = '';
			END IF;
		END IF;
	END IF;
END//
DELIMITER ;

/*manual license activation*/
DELIMITER //
CREATE PROCEDURE ManualActivateLicense (IN appGuid VARCHAR(40), IN sn VARCHAR(60), IN regKey
VARCHAR(33), IN hardwareSignature INT, IN notes TEXT, IN alias VARCHAR(50), IN pass VARCHAR(50),
OUT ret VARCHAR(500))
BEGIN
	DECLARE smid INT;
	DECLARE regDescId INT;
	DECLARE softId INT;
	DECLARE userLicCode INT;
	DECLARE userFeatures INT;
	DECLARE userExpTime DATETIME;
	SET ret = 'Fail;Unknown error';
	SELECT SoftwareManagerID into smid FROM softwareManager WHERE ManagerAlias=alias AND
PASSWORD(pass) = ManagerPass AND (ManagerLevel = NULL or ManagerLevel = 0);
	if smid IS NULL THEN
		SET ret = 'Fail;Login error';
	ELSE
		IF Exists(SELECT 1 FROM softwareRegister WHERE RegisterKey=regKey) THEN
			SET ret = 'Fail;registry already used';
		ELSE
			SELECT
				softwareLicense.SoftwareID,
				softwareLicense.InstallingLicenseCode,
				softwareLicense.InstallingFeatures,
				softwareLicense.InstallingExpireTime
			INTO softId,userLicCode,userFeatures,userExpTime
			FROM (softwareLicense INNER JOIN softwareProtected
				ON softwareLicense.SoftwareID=softwareProtected.SoftwareID)
			LEFT JOIN softwareRegister
				ON softwareLicense.RegisterID = softwareRegister.RegisterID
			WHERE softwareprotected.AppGuid = appGuid AND softwarelicense.SerialNumber
= sn;
			IF softId IS NULL THEN
				SET ret = 'Fail;invalid serial number';
```

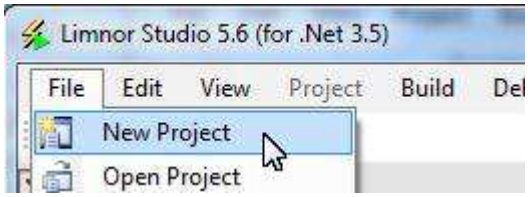
```
                        ELSE
                                INSERT INTO softwareregisterdesc (RegisterType,RegisterDescription)
VALUES (2,notes);
                                SELECT (SELECT LAST_INSERT_ID()) into regDescId;
                                INSERT softwareRegister (
                                        SoftwareID,
                                        RegisterTime,
                                        SerialNumber,
                                        HardwareSignature,
                                        RegisterKey,
                                        LicenseCode,
                                        Features,
                                        ExpireTime,
                                        SoftwareManagerID,
                                        RegisterDescID
                                ) VALUES (
                                        softId,
                                        (SELECT NOW()),
                                        sn,
                                        hardwareSignature,
                                        regKey,
                                        userLicCode,
                                        userFeatures,
                                        userExpTime,
                                        smid,
                                        regDescId
                                );
                                UPDATE softwareLicense SET RegisterID=(SELECT LAST_INSERT_ID())
WHERE SoftwareID=softId AND SerialNumber=sn;
                                SET ret = CONCAT('OK;',CAST(userLicCode AS CHAR), ';',
CAST(userFeatures AS CHAR), ';', IFNULL(DATE_FORMAT(userExpTime, '%Y-%m-%d'),''));
                        END IF;
                END IF;
        END IF;
END//
DELIMITER ;

/*manually deactivate*/
DELIMITER //
CREATE PROCEDURE ManualDeactivateLicense (IN appGuid VARCHAR(40), IN sn VARCHAR(60), IN regKey
VARCHAR(33), IN notes TEXT, IN alias VARCHAR(50), IN pass VARCHAR(50), OUT ret VARCHAR(500))
BEGIN
        DECLARE smid INT;
        DECLARE softId INT;
        DECLARE fs INT;
        DECLARE hd INT;
        DECLARE expTime DATETIME;
        DECLARE regDescId INT;
        DECLARE regId INT;
        SET ret = 1;
        SELECT SoftwareManagerID into smid FROM softwareManager WHERE ManagerAlias=alias AND
PASSWORD(pass) = ManagerPass AND (ManagerLevel = NULL or ManagerLevel = 0);
        if smid IS NULL THEN
                SET ret = 'Login error';
        ELSE
                SELECT softwareLicense.SoftwareID,

        softwareRegister.Features,softwareRegister.ExpireTime,softwareRegister.HardwareSignature
                        into softId, fs,  expTime, hd
                FROM (softwareLicense INNER JOIN softwareProtected    ON
softwareLicense.SoftwareID=softwareProtected.SoftwareID)       LEFT JOIN softwareRegister   ON
softwareLicense.RegisterID = softwareRegister.RegisterID
                WHERE softwareprotected.AppGuid = appGuid AND softwarelicense.SerialNumber = sn;
                if softId IS NULL THEN
                        SET ret = 'Invalid serial number';
                ELSE
                        INSERT INTO softwareregisterdesc (RegisterType,RegisterDescription) VALUES
(3,notes);
                        SELECT (SELECT LAST_INSERT_ID()) into regDescId;
                        INSERT INTO softwareRegister (
                                SoftwareID,
```

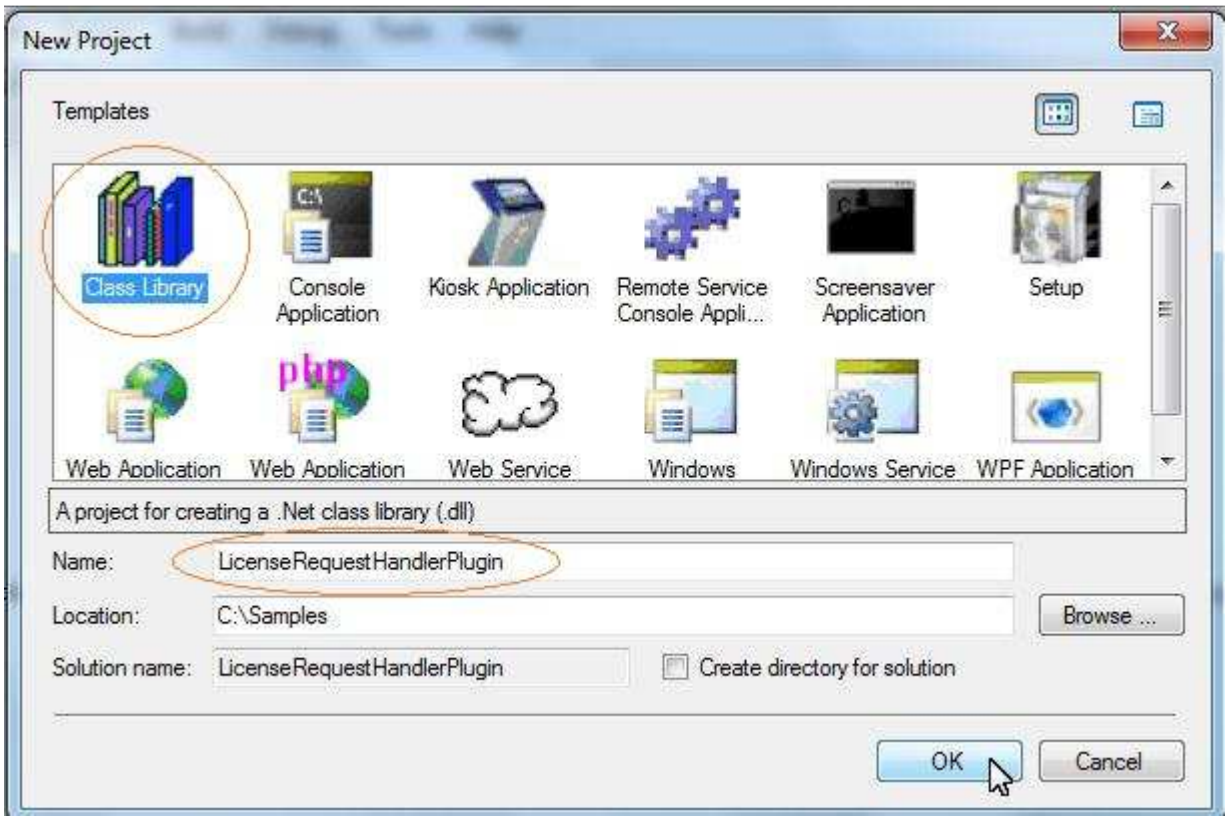
```
                                RegisterTime,
                                SerialNumber,
                                HardwareSignature,
                                LicenseCode,
                                Features,
                                ExpireTime,
                                RegisterKey,
                                SoftwareManagerID,
                                RegisterDescID)
                        VALUES (
                                softId,
                                (SELECT NOW()),
                                sn,
                                hd,
                                0,
                                fs,
                                expTime,
                                regKey,
                                smid,
                                regDescId);
                        UPDATE softwareLicense SET RegisterID=(SELECT LAST_INSERT_ID()) WHERE
SoftwareID=softId AND SerialNumber=sn;
                        SET ret = '';
                END IF;
        END IF;
END//
DELIMITER ;

/*create a serial number*/
DELIMITER //
CREATE PROCEDURE AddSerialNumber (IN appGuid VARCHAR(40), IN sn VARCHAR(60), IN licenseCode INT,
IN features INT, IN expireTime DATETIME)
BEGIN
        DECLARE softId INT;
        SELECT SoftwareID INTO softId FROM softwareProtected WHERE AppGuid = appGuid;
        IF softId IS NULL THEN
                SELECT NULL as SoftwareID;
        ELSE
                INSERT INTO softwareLicense (
                        SoftwareID,
                        SerialNumber,
                        CreateTime,
                        InstallingLicenseCode,
                        InstallingFeatures,
                        InstallingExpireTime
                        )
                VALUES (
                        softId,
                        sn,
                        (SELECT NOW()),
                        licenseCode,
                        features,
                        expireTime
                        );
                SELECT softId as SoftwareID;
     END IF;
END//
DELIMITER ;

/*end of installation*/
```

Create License Management Plug-in Project

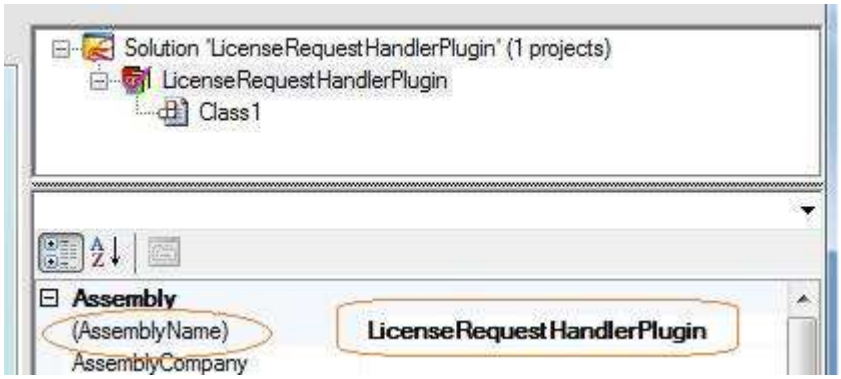
- A license management plug-in must be a DLL named "**LicenseRequestHandlerPlugin.dll**".
- The DLL must contain a public class which implements ILicenseRequestHandler interface.

Create a new class library project:





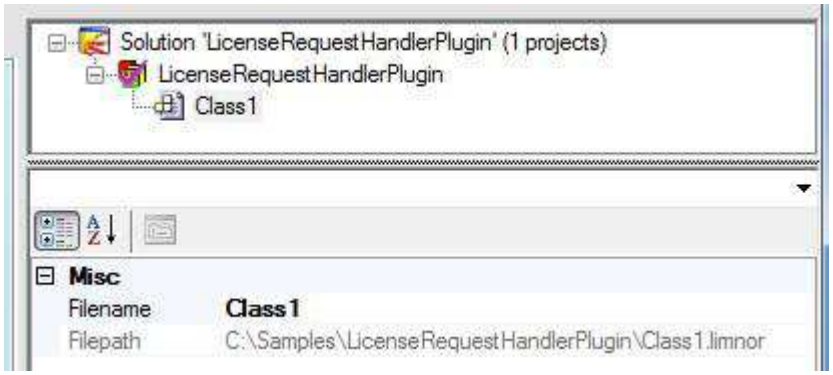
A class library project is created:



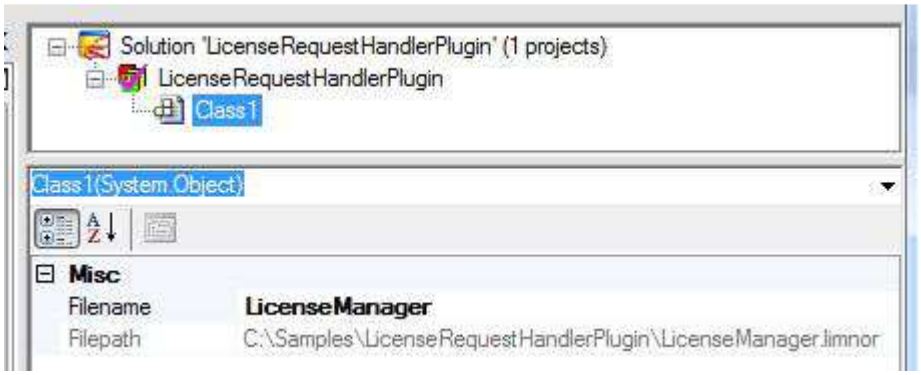
Note that the AssemblyName of the project must be “LicenseRequestHandlerPlugin”. Do not change it.

Create License Management Plug-in Class

The project created a class named Class1:

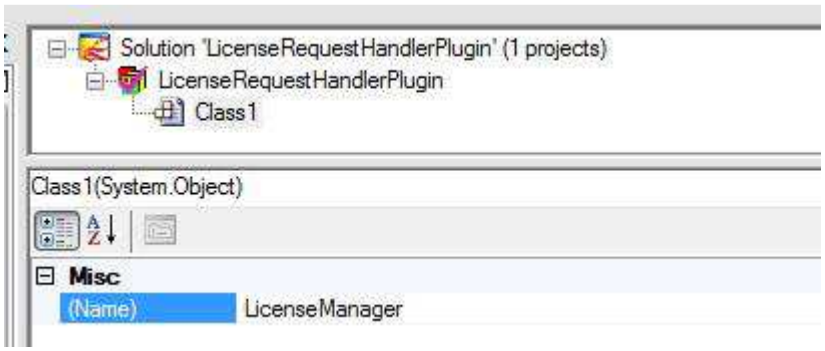


We may change the class name. Let's name it LicenseManager. First, change its file name:



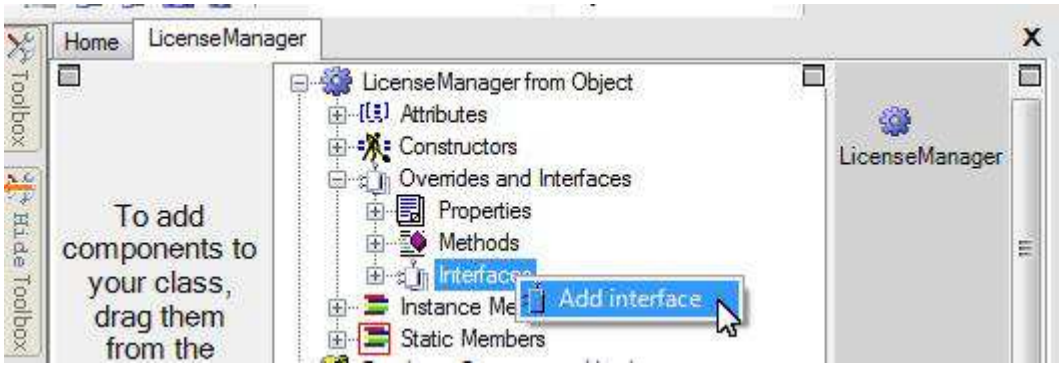
Open it into its designer:

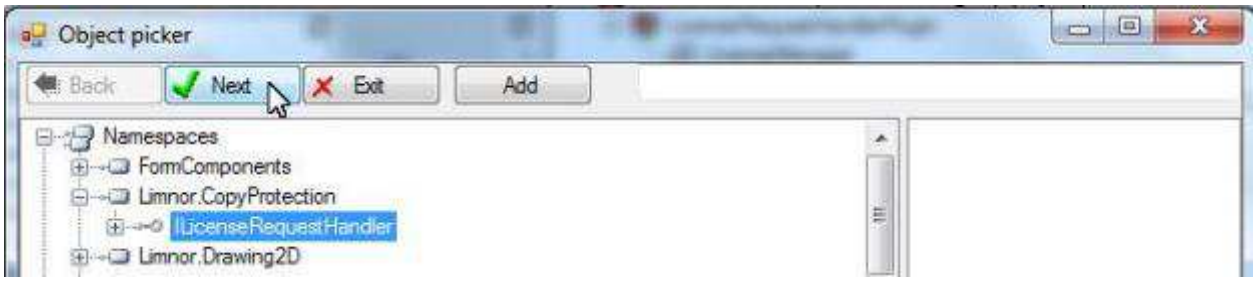
Change class name to LicenseManager:



Implement Interface ILicenseRequestHandler

For the LicenseManager class to be a license management plug-in, we need to implement interface ILicenseRequestHandler.





On adding the interface ILicenseRequesthandler, we can see the interface has 4 methods:

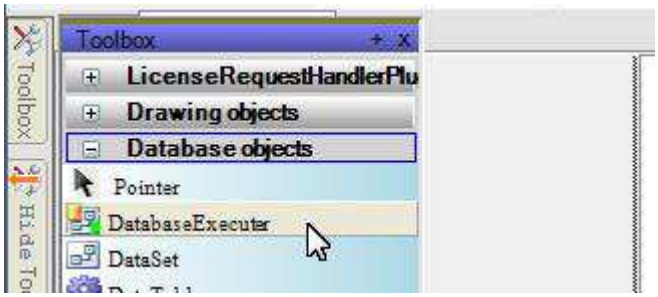
We are going to edit each method according to our license management logics.

Implement OnHandleLicenseRequest

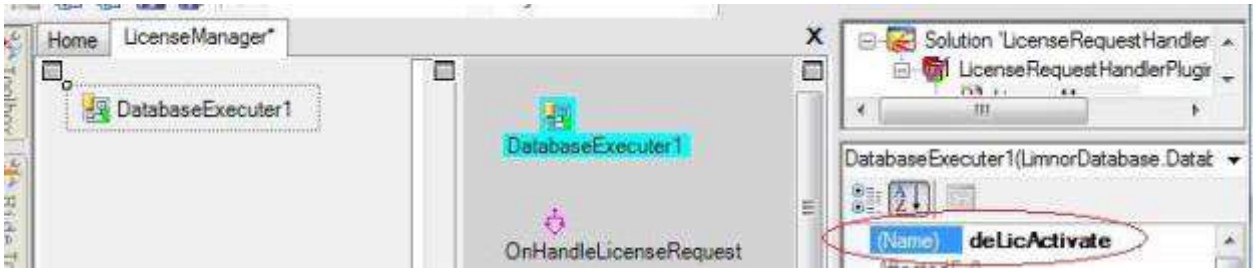
Web page RegProc.html calls this method to request activation of a license. This method tells the web page whether a license should be issued and how the license should be generated.

Use DatabaseExecuter

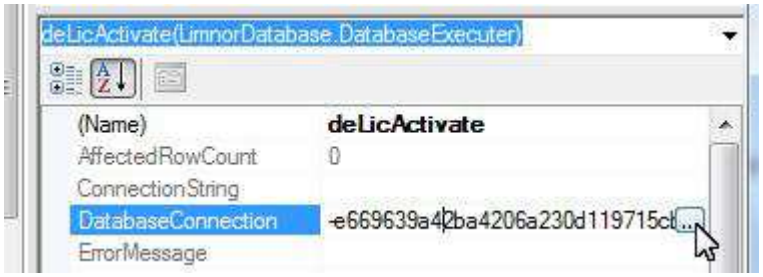
Stored-procedure ActivateLicense implements our sample handling of license activation. We'll use a DatabaseExecuter component to execute this stored-procedure.



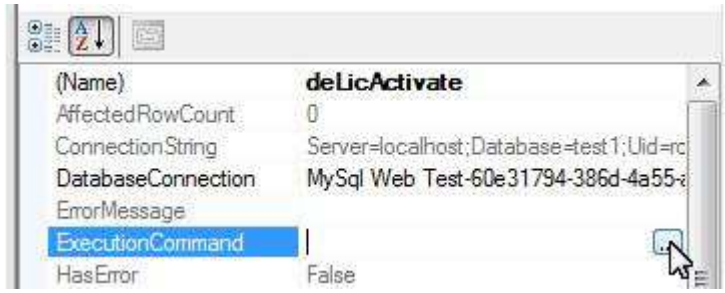
Rename it to deLicActivate:



Set its database connection:



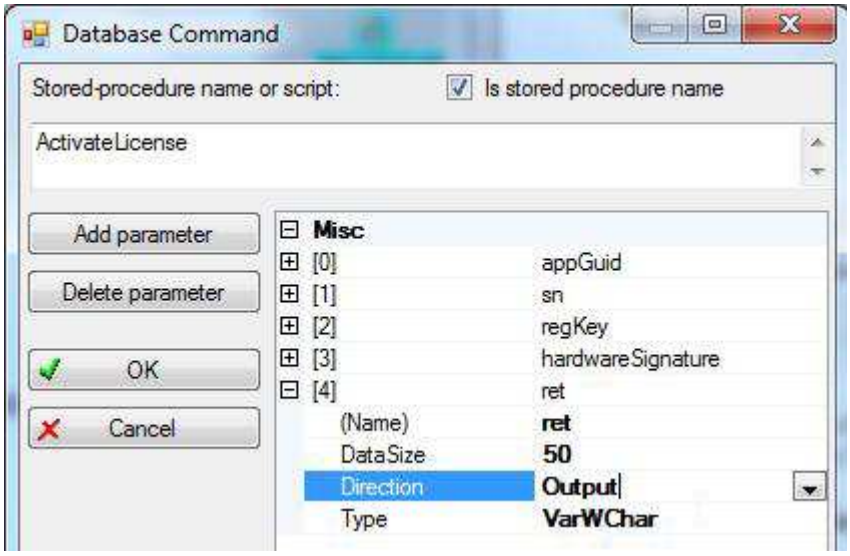
Set its ExecutionCommand:



Set stored-procedure name and parameters according to the script for creating database objects:

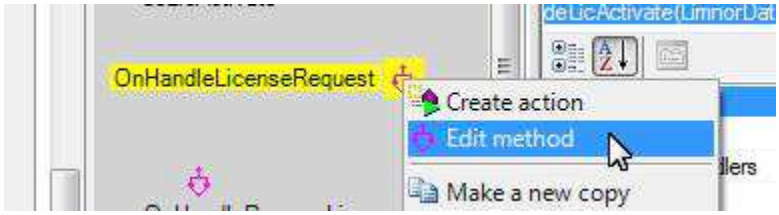
```
CREATE PROCEDURE ActivateLicense (IN appGuid VARCHAR(40), IN sn VARCHAR(60), IN regKey
VARCHAR(33), IN hardwareSignature INT, OUT ret VARCHAR(50))
```

Note that the last parameter is an output parameter. We need to change its Direction property to reflect it:

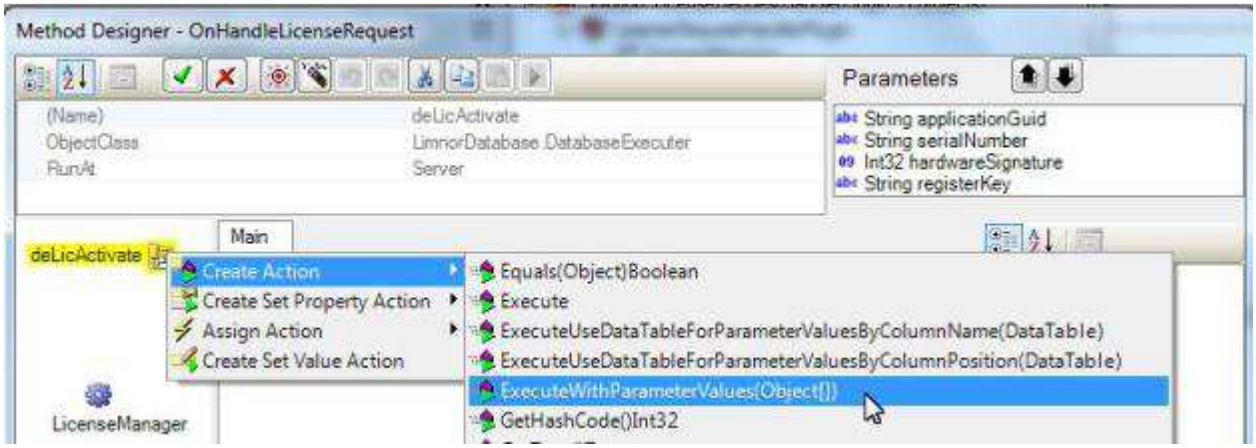


Modify OnHandleLicenseRequest

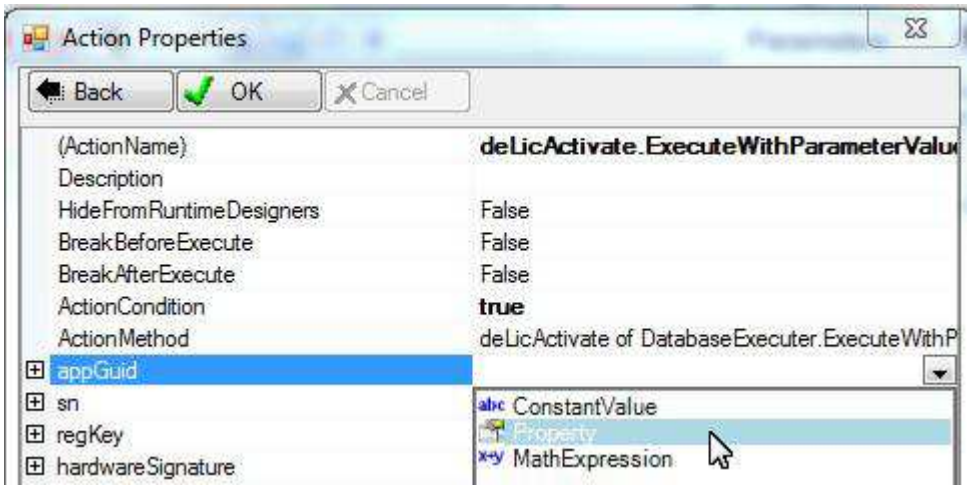
Now we may edit method OnHandleLicenseRequest:

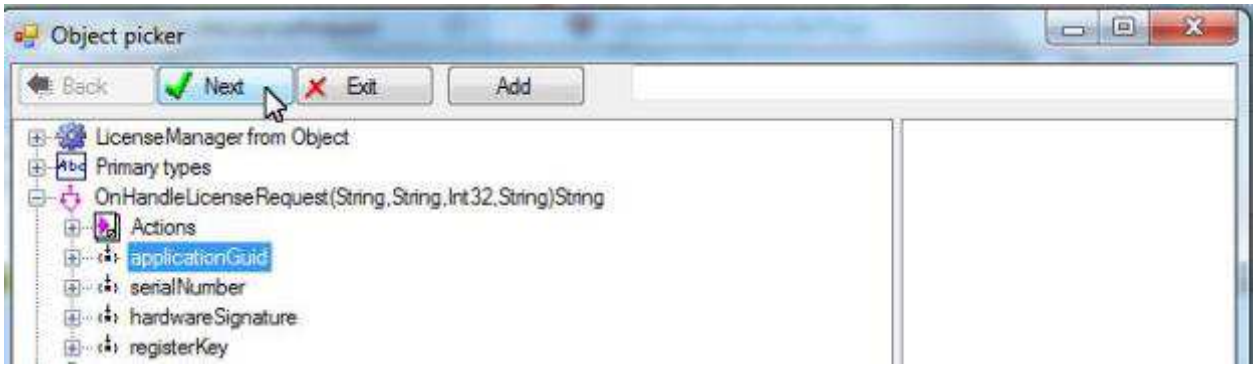


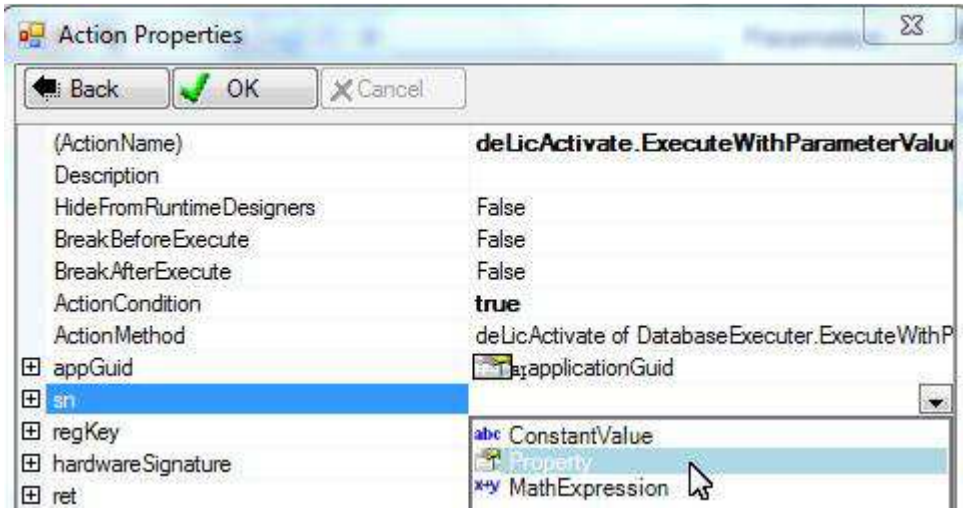
Create an action to execute license activation stored-procedure:



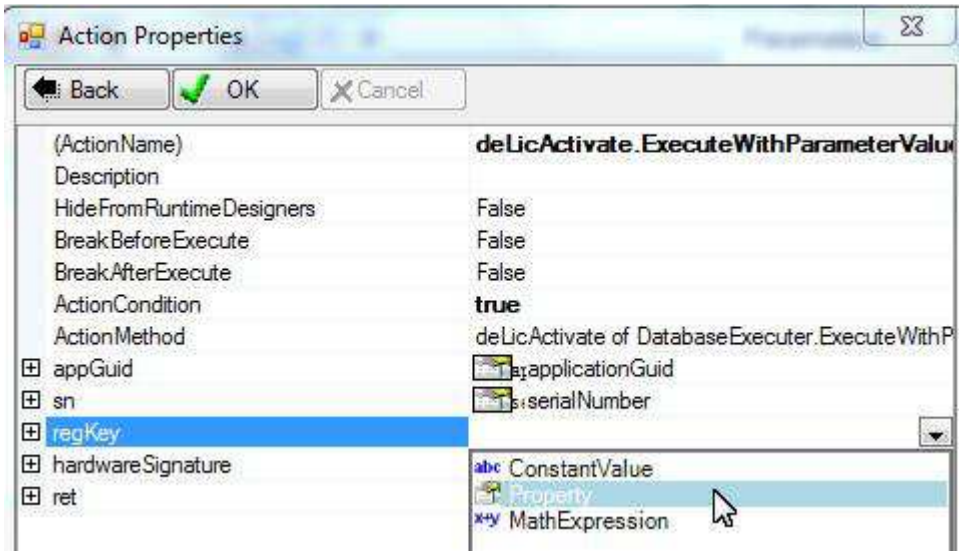
Pass method parameters to the action:

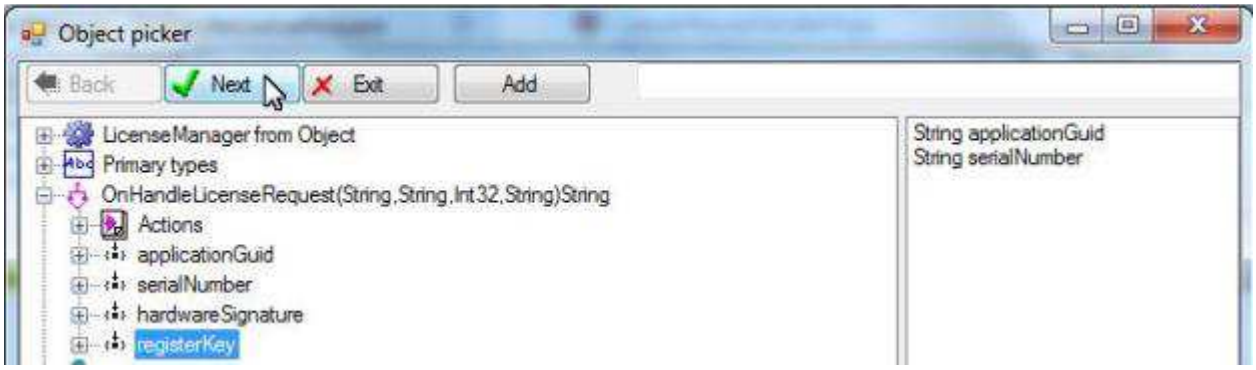

Action Properties
Back
OK
Cancel
(ActionName)
deLicActivate.ExecuteWithParameterValu
Description
HideFromRuntimeDesigners
False
BreakBeforeExecute
False
BreakAfterExecute
False
ActionCondition
true
ActionMethod
deLicActivate of DatabaseExecuter.ExecuteWithP
appGuid
sn
regKey
hardwareSignature
ConstantValue
Property
MathExpression

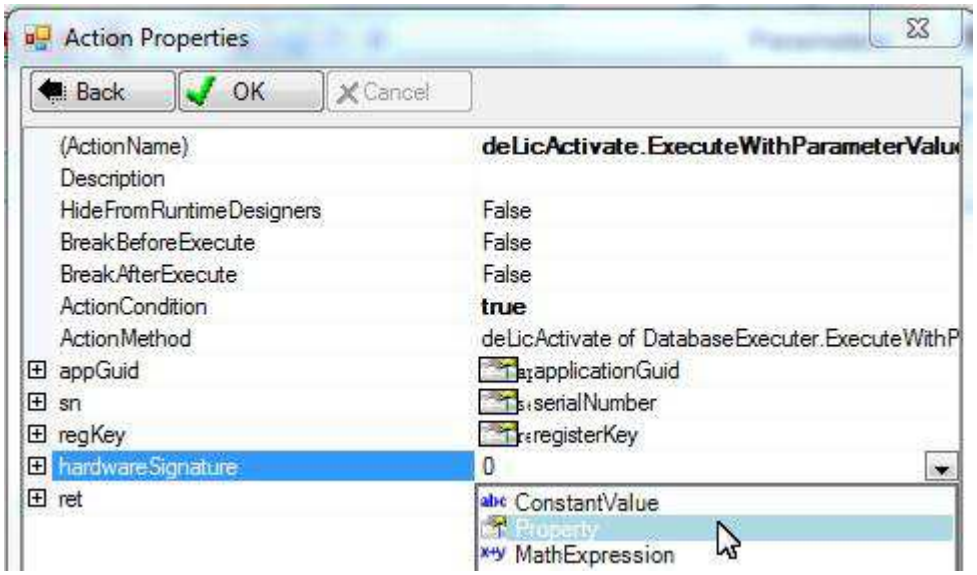

Object picker
Back
Next
Exit
Add
LicenseManager from Object
Primary types
OnHandleLicenseRequest(String,String,Int32,String)String
Actions
applicationGuid
serialNumber
hardwareSignature
registerKey

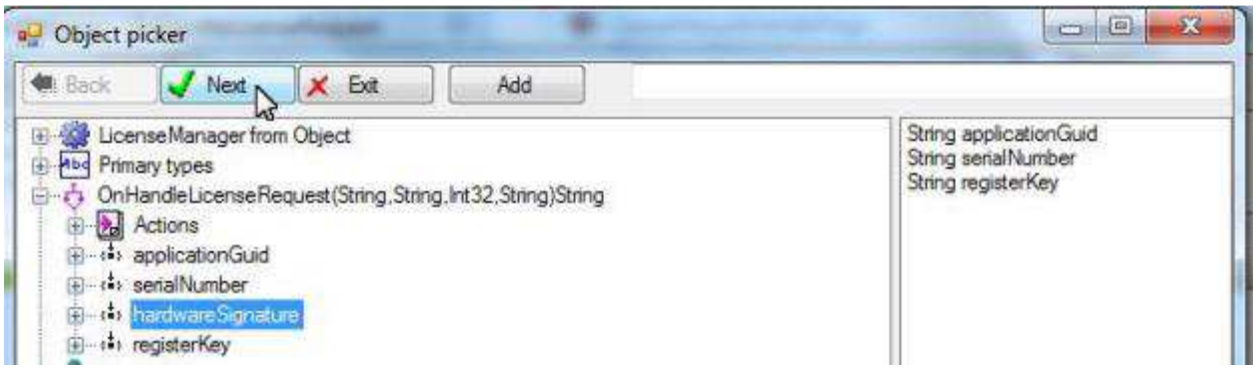

Action Properties
Back
OK
Cancel
(ActionName)
deLicActivate.ExecuteWithParameterValu
Description
HideFromRuntimeDesigners
False
BreakBeforeExecute
False
BreakAfterExecute
False
ActionCondition
true
ActionMethod
deLicActivate of DatabaseExecuter.ExecuteWithP
appGuid
applicationGuid
sn
regKey
hardwareSignature
ret
ConstantValue
Property
MathExpression


Object picker
Back
Next
Exit
Add
LicenseManager from Object
Primary types
OnHandleLicenseRequest(String,String,Int32,String)String
Actions
applicationGuid
serialNumber
hardwareSignature
registerKey
String applicationGuid

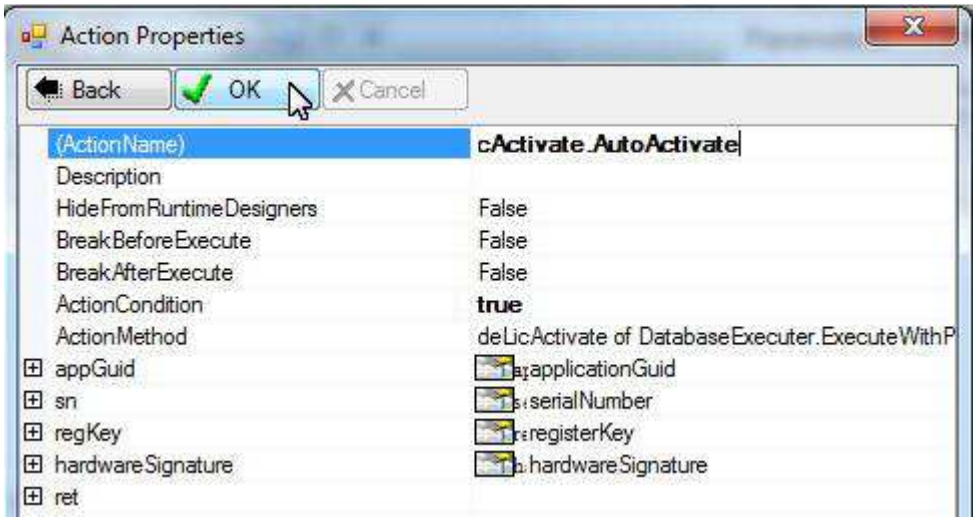

Action Properties
Back
OK
Cancel
(ActionName)
deLicActivate.ExecuteWithParameterValu
Description
HideFromRuntimeDesigners
False
BreakBeforeExecute
False
BreakAfterExecute
False
ActionCondition
true
ActionMethod
deLicActivate of DatabaseExecuter.ExecuteWithP
appGuid
applicationGuid
sn
serialNumber
regKey
hardwareSignature
ret
ConstantValue
Property
MathExpression


Object picker
Back
Next
Exit
Add
LicenseManager from Object
Primary types
OnHandleLicenseRequest(String,String,Int32,String)String
Actions
applicationGuid
serialNumber
hardwareSignature
registerKey
String applicationGuid
String serialNumber

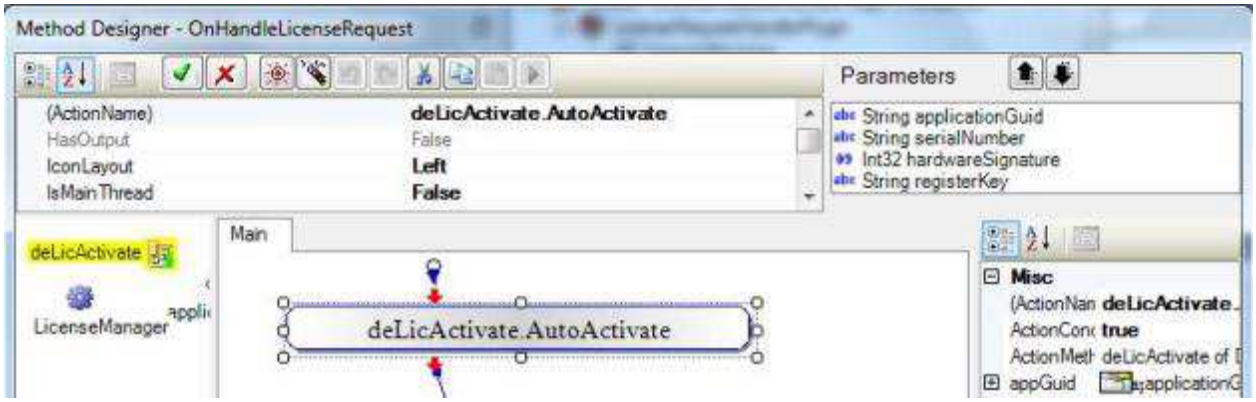




Rename the action and click OK.



The action is created and appears in the Action Pane:



This interface method, OnHandleLicenseRequest, returns a string to notify the caller how license activation is processed. The return value is a semi-colon delimited string.

If a license can be activated then the return value should be formed by “OK”, license code, an integer representing allowed features and expiration time. For example:

OK;1;3;2012-12-27

The above return value indicates that it is OK to issue a license code 1, features 3 and expiration time 2012-12-27. But since the license code is 1, the expiration time is not used.

License Code	Meaning
1	No expiration. Hardware signature not checked
2	Use expiration. Hardware signature not checked
3	No expiration. Hardware signature is checked
4	Use expiration. Hardware signature is checked

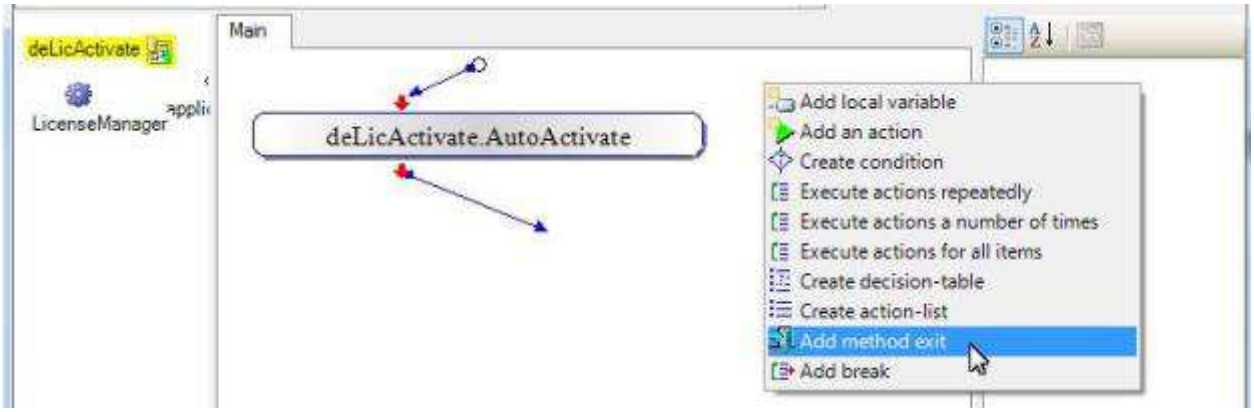
The caller uses the information to issue a license.

If a license cannot be activated then the method should return “Fail” followed by an error message. For example:

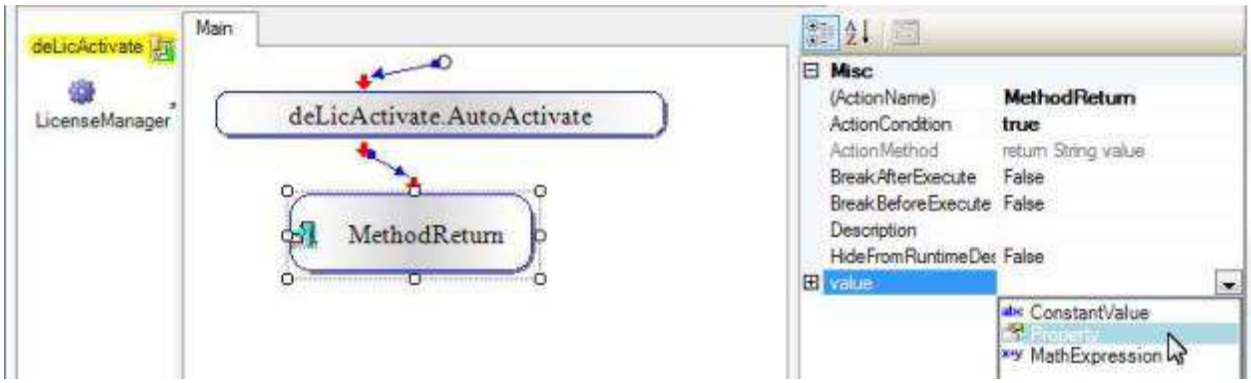
Fail;Invalid serial number

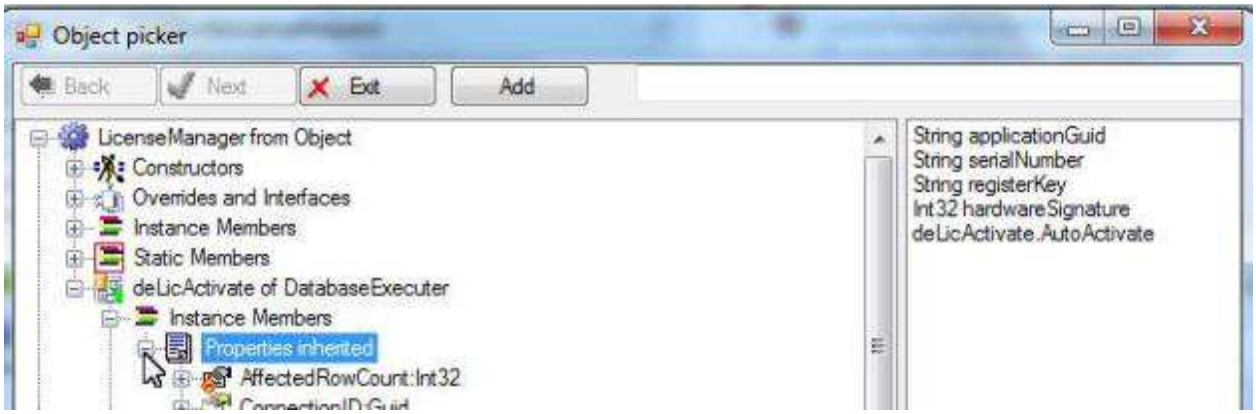
The caller will pass the error message to the user.

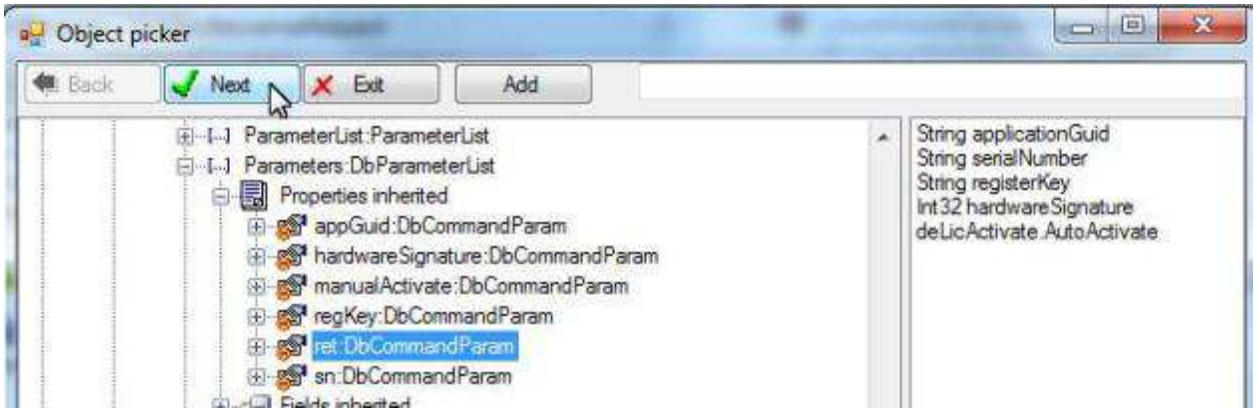
Because the stored-procedure already returns a string in the above format, we may simply add an action to return it:



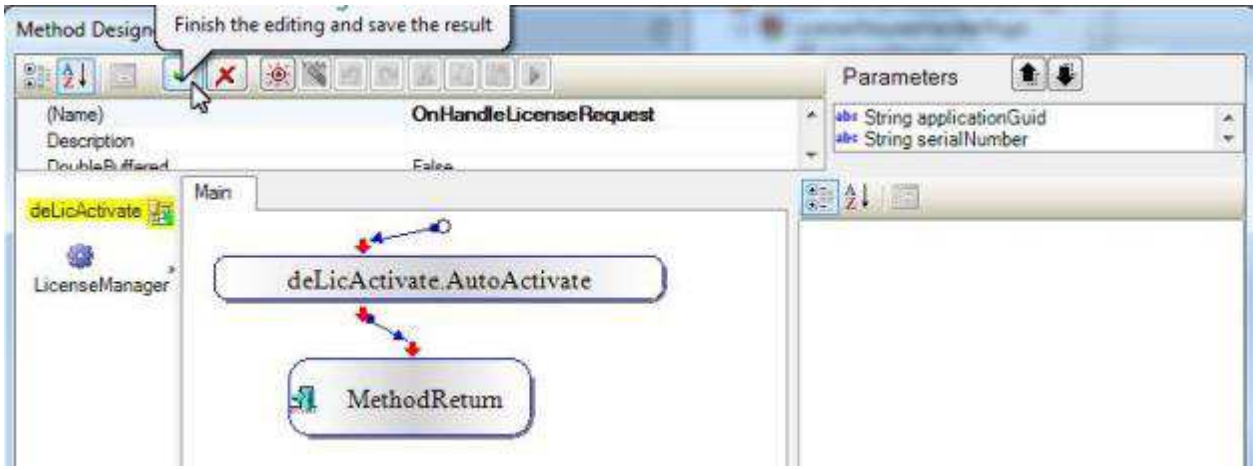
Link the new action to the last action. Set its return value to parameter “ret” of the database executer:







That is all we need for this sample:

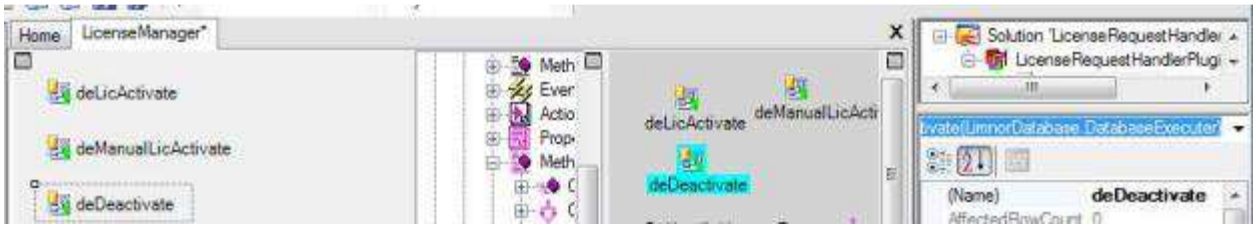


Implement OnHandleRemoveLicenseRequest

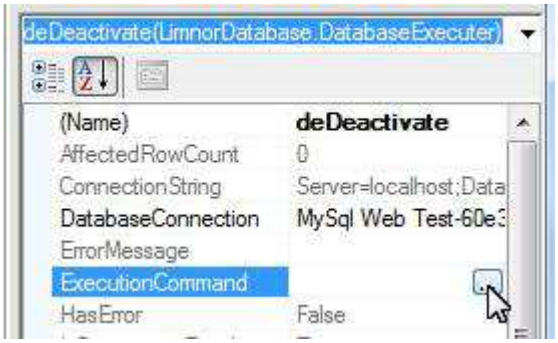
Web page RegProc.html calls this method to request a deactivation of a license. This method checks that the request is indeed for the current installed license. This method returns an empty string to indicate that the license is deactivated. This method returns an error message if it rejects the request.

Use Database Executer

Add a new DatabaseExecuter to the class and rename it to deDeactivate:



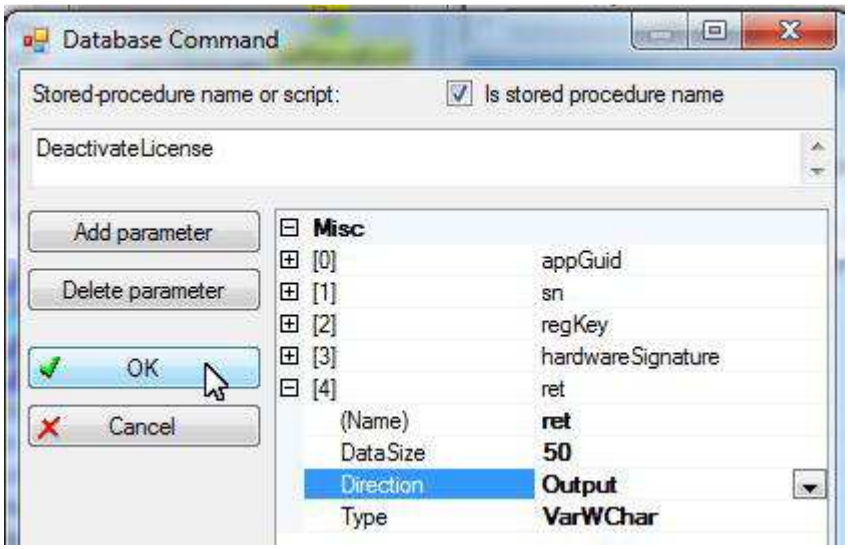
Set its ExecutionCommand:



Set stored-procedure name and parameters according to the script for creating database objects:

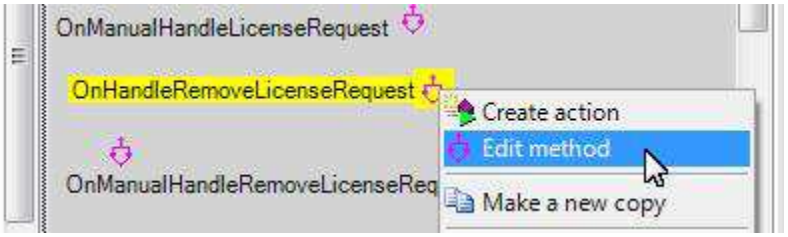
```
CREATE PROCEDURE DeactivateLicense (IN appGuid VARCHAR(40), IN sn VARCHAR(60), IN regKey
VARCHAR(33), IN hardwareSignature INT, OUT ret VARCHAR(50))
```

Note that the last parameter is an output parameter. We need to change its Direction property to reflect it:

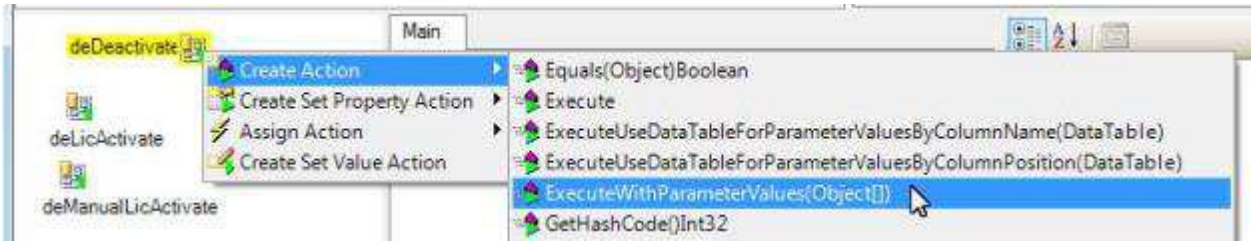


Modify OnHandleRemoveLicenseRequest

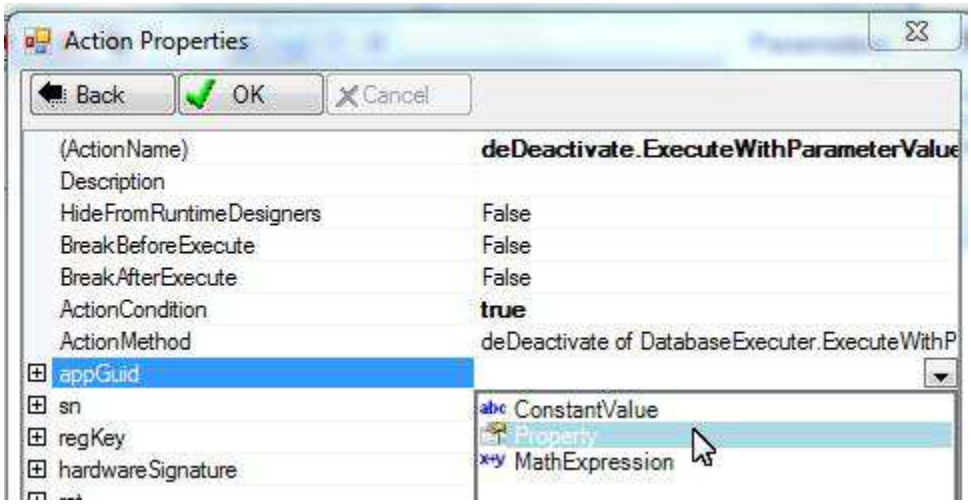
Now we may edit method OnHandleRemoveLicenseRequest

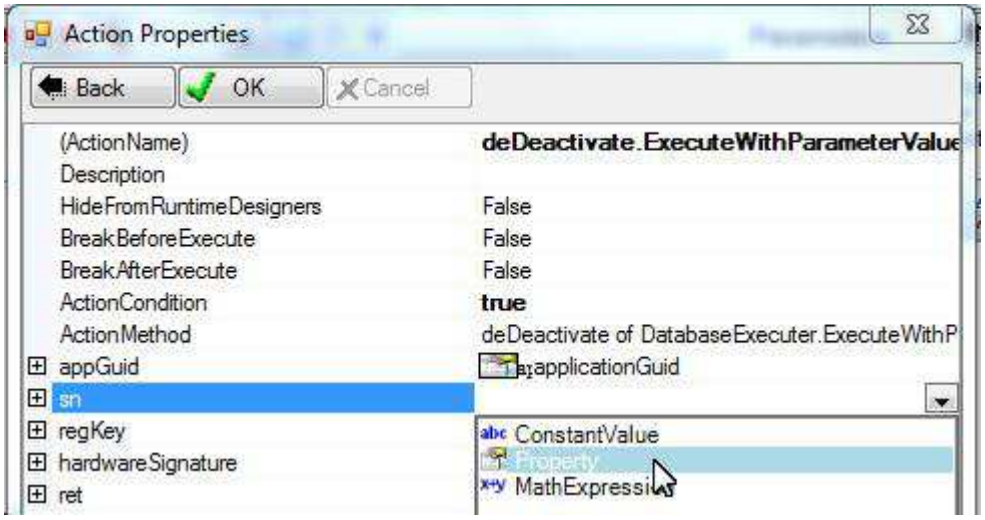


Create an action to execute license deactivation stored-procedure:

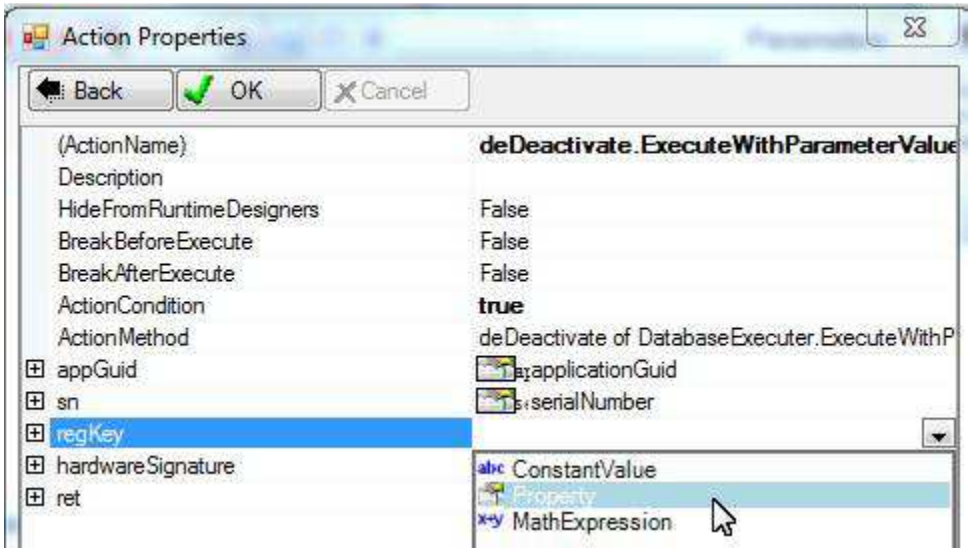


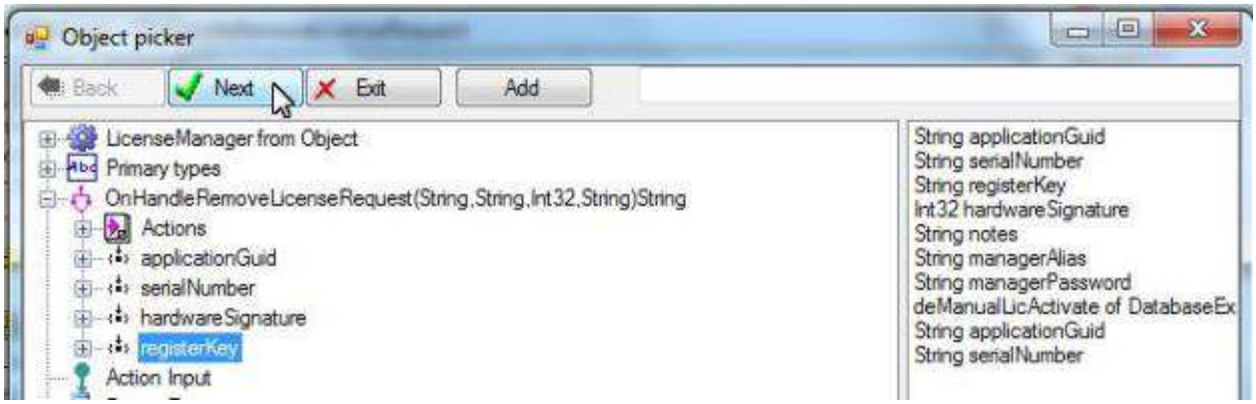
Pass method parameters to the action:

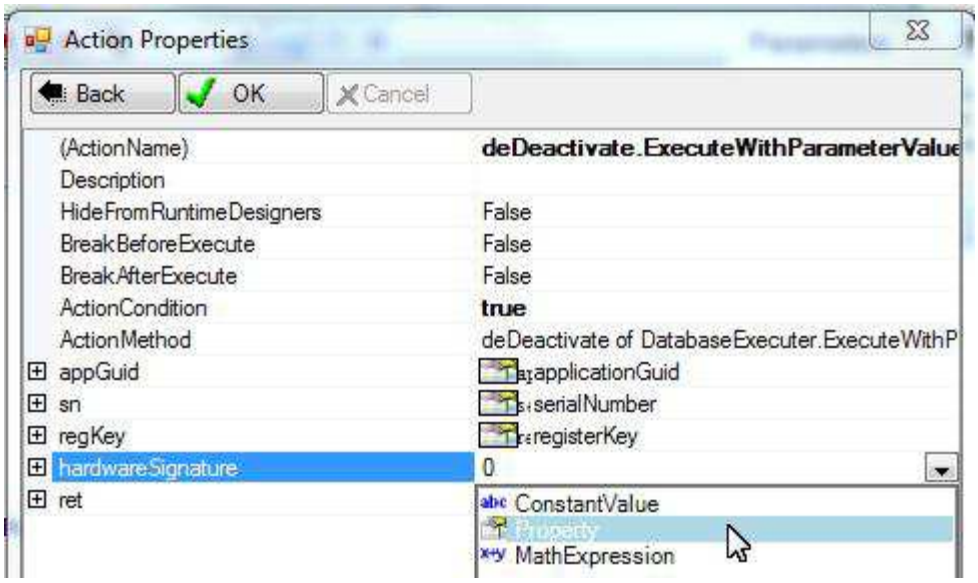


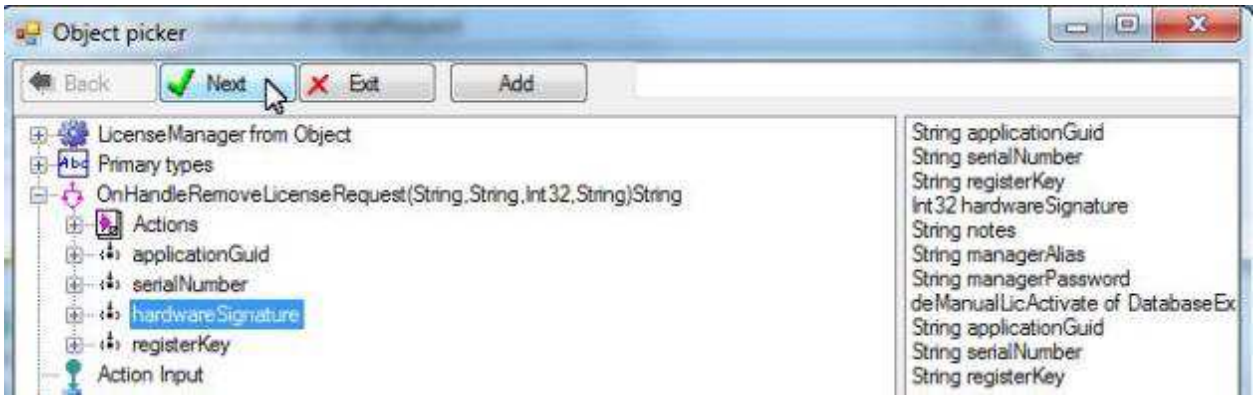

Action Properties
Back
OK
Cancel
(ActionName)
deDeactivate.ExecuteWithParameterValue
Description
HideFromRuntimeDesigners
False
BreakBeforeExecute
False
BreakAfterExecute
False
ActionCondition
true
ActionMethod
deDeactivate of DatabaseExecuter.ExecuteWithP
appGuid
applicationGuid
sn
regKey
hardwareSignature
ret
ConstantValue
Property
MathExpressi


Object picker
Back
Next
Exit
Add
LicenseManager from Object
Primary types
OnHandleRemoveLicenseRequest(String,String,Int32,String)String
Actions
applicationGuid
serialNumber
hardwareSignature
String applicationGuid
String serialNumber
String registerKey
Int32 hardwareSignature
String notes
String managerAlias
String managerPassword
deManualLicActivate of DatabaseEx

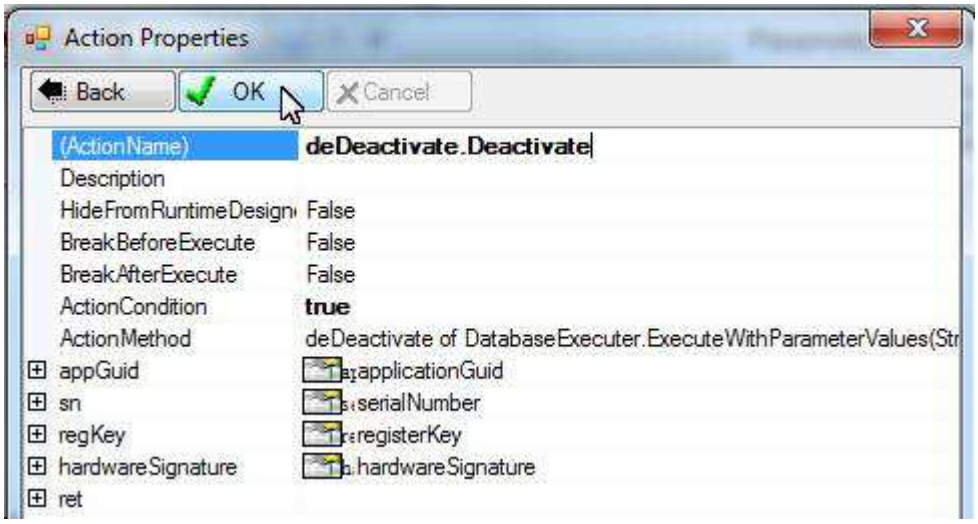

Action Properties
Back
OK
Cancel
(ActionName)
deDeactivate.ExecuteWithParameterValue
Description
HideFromRuntimeDesigners
False
BreakBeforeExecute
False
BreakAfterExecute
False
ActionCondition
true
ActionMethod
deDeactivate of DatabaseExecuter.ExecuteWithP
appGuid
applicationGuid
sn
serialNumber
regKey
hardwareSignature
ret
ConstantValue
Property
MathExpression



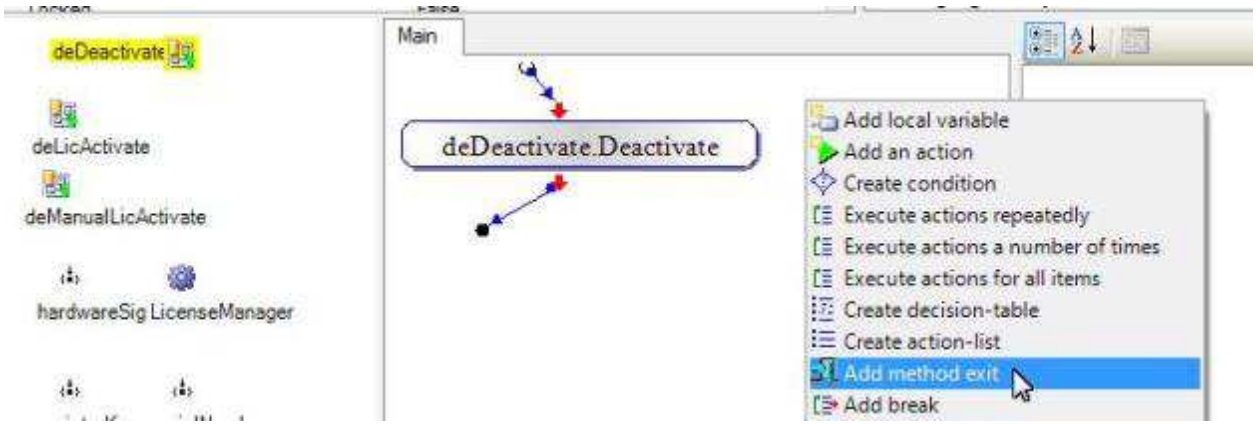


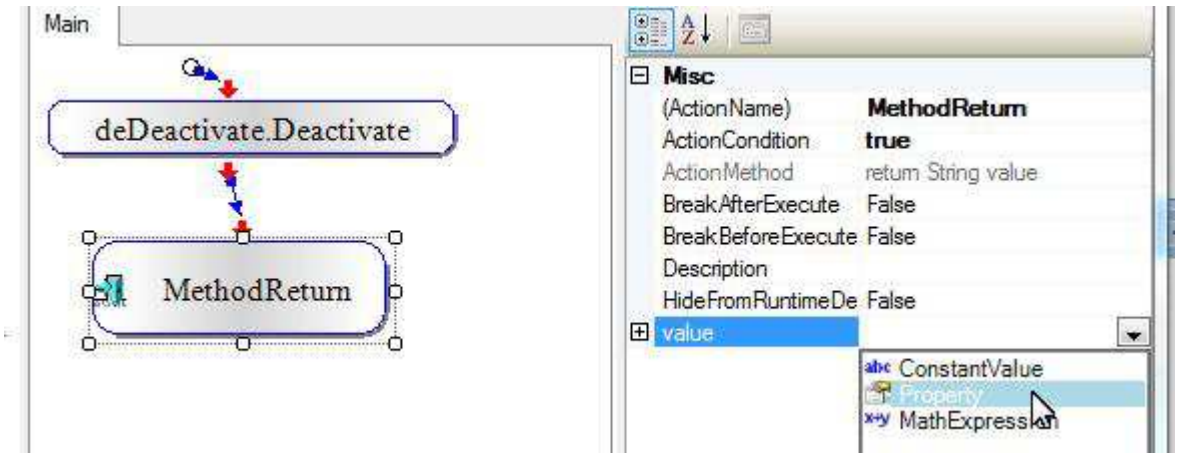


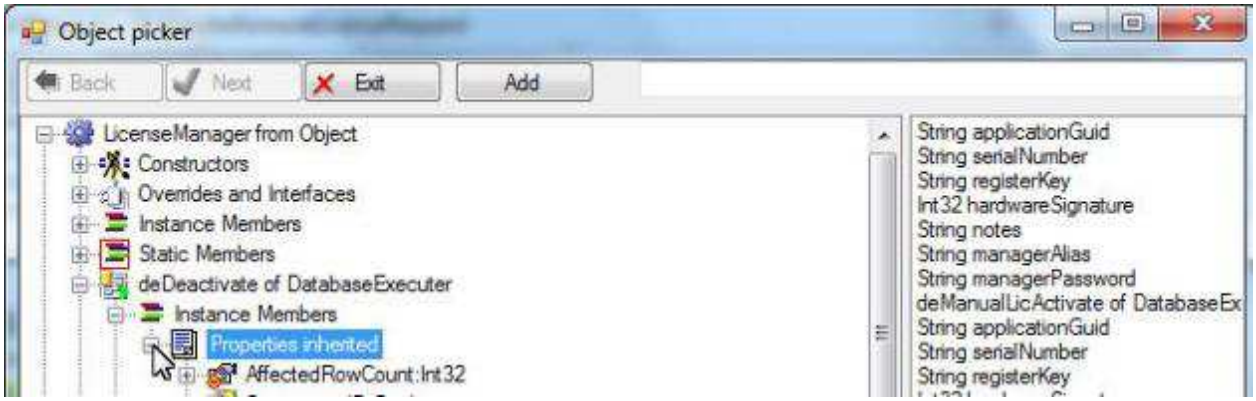
Rename the action and click OK:

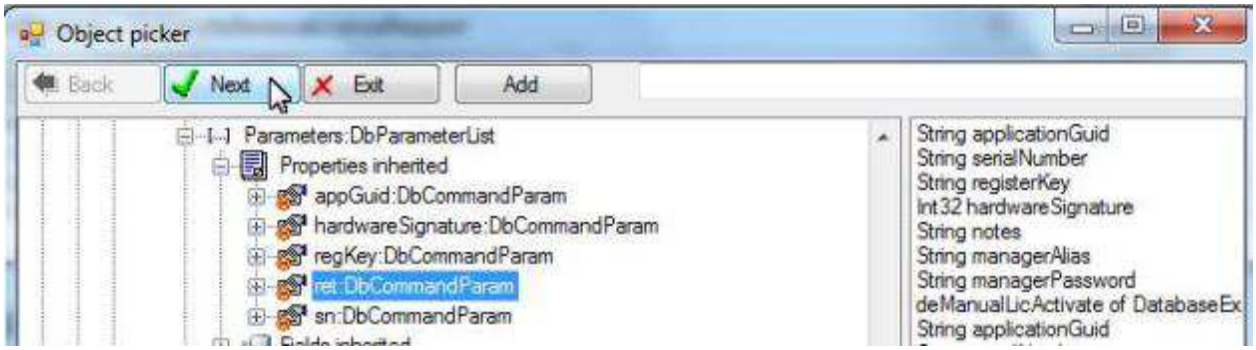


Add a method return action to return parameter “ret” of the DatabaseExecuter:

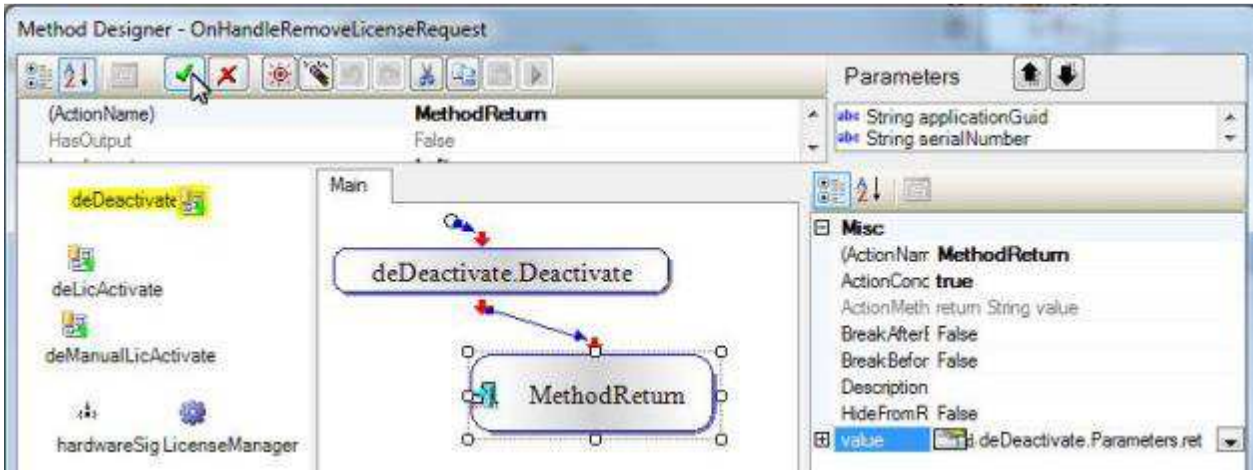








We are done modifying this method.

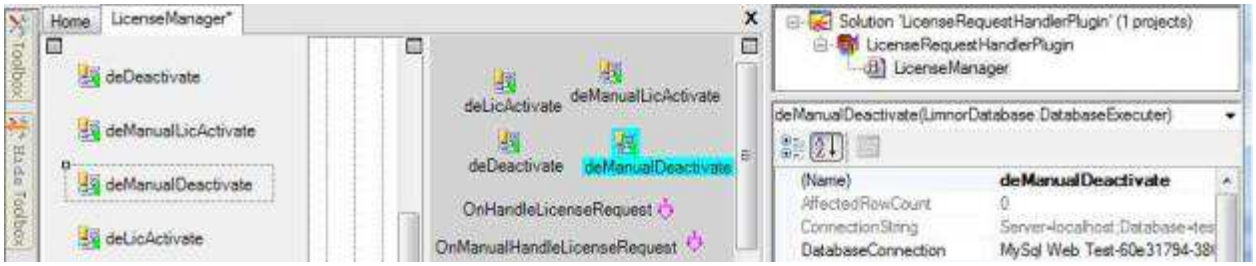


Implement OnManualHandleRemoveLicenseRequest

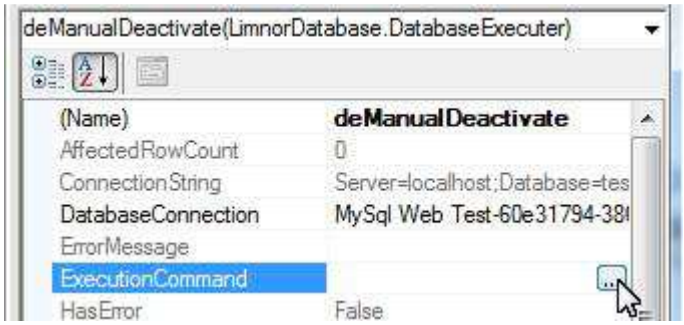
Web page Deactivate.html calls this method to request an unconditional license deactivation.

Use DatabaseExecuter

Add new DatabaseExecuter and name it deManualDeactivate.



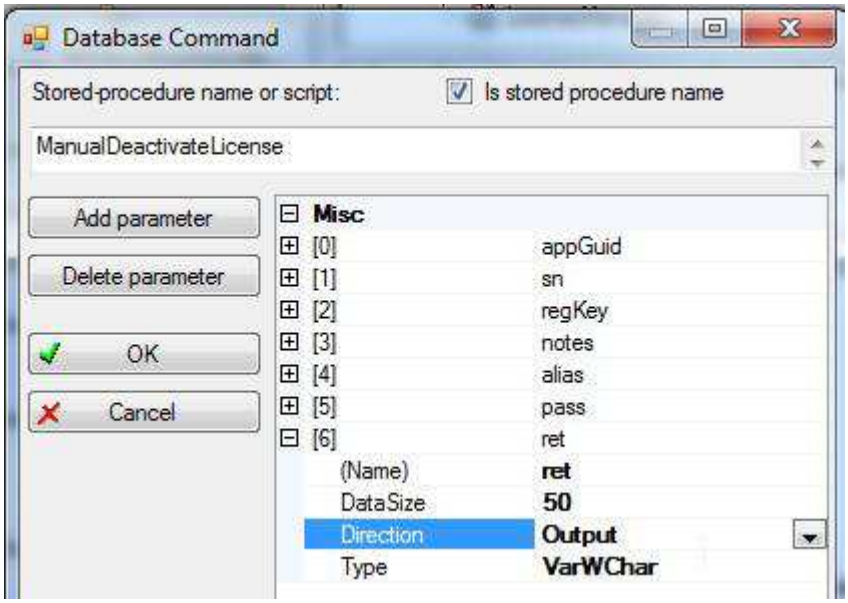
Set its ExecutionCommand:



Set stored-procedure name and parameters according to the script for creating database objects:

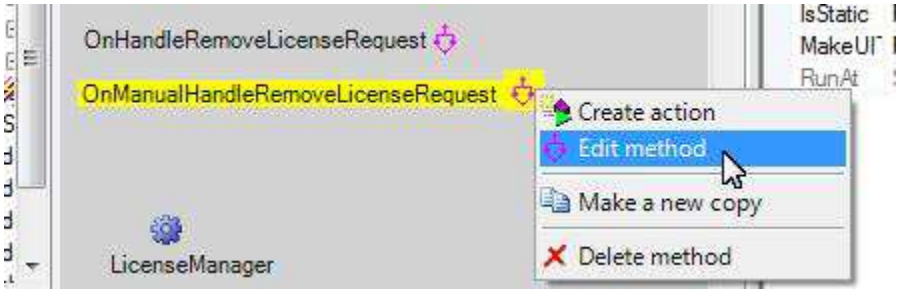
```
CREATE PROCEDURE ManualDeactivateLicense (IN appGuid VARCHAR(40), IN sn VARCHAR(60), IN regKey
VARCHAR(33), IN notes TEXT, IN alias VARCHAR(50), IN pass VARCHAR(50), OUT ret VARCHAR(50))
```

Note that the last parameter is an output parameter. We need to change its Direction property to reflect it:



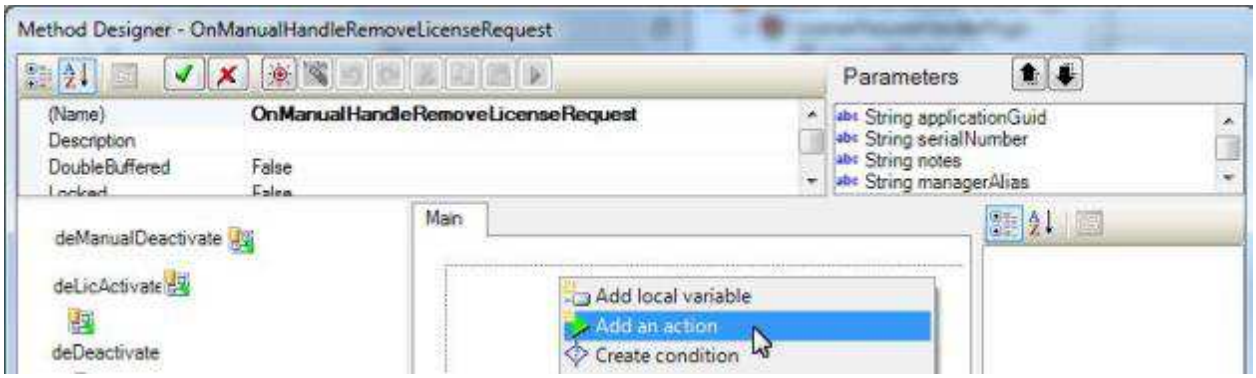
Modify OnManualHandleRemoveLicenseRequest

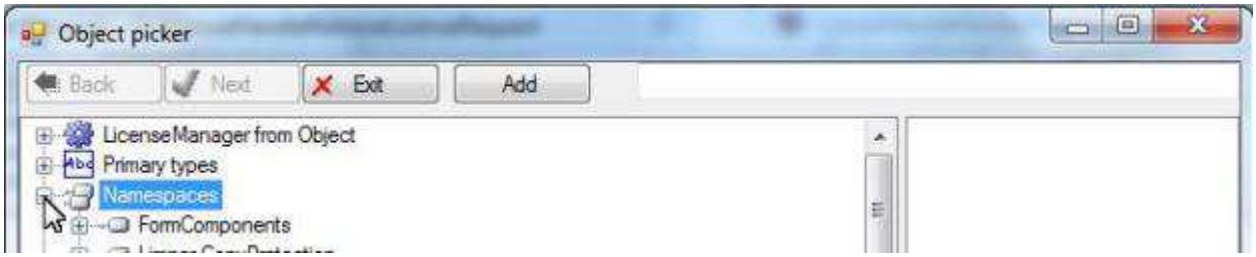
Now we may edit method OnManualHandleRemoveLicenseRequest:

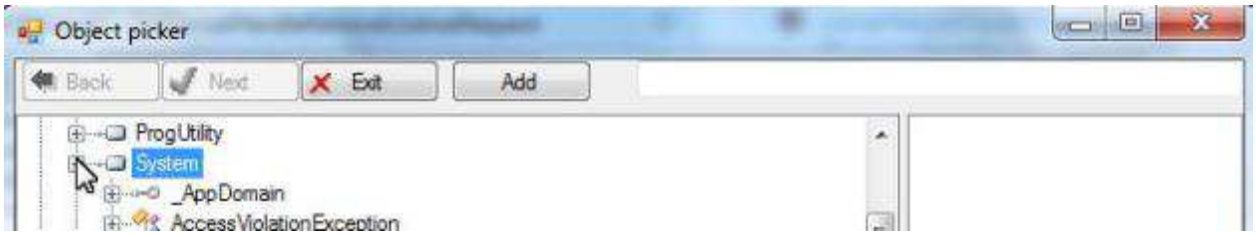


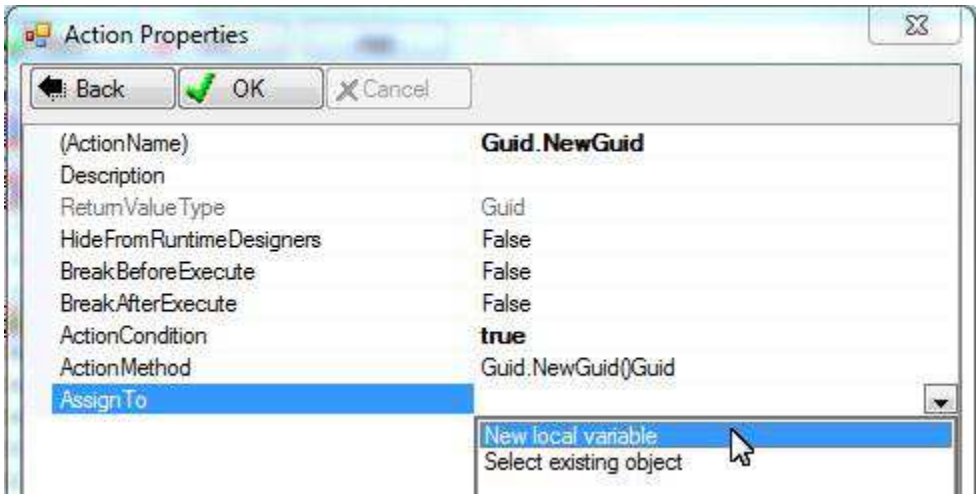
Create a new registry key

Because it is an unconditional deactivation, we do not know client register key. We just generate a new key for it. A new GUID can be created by executing NewGuid method. Let's create a NewGuid action:

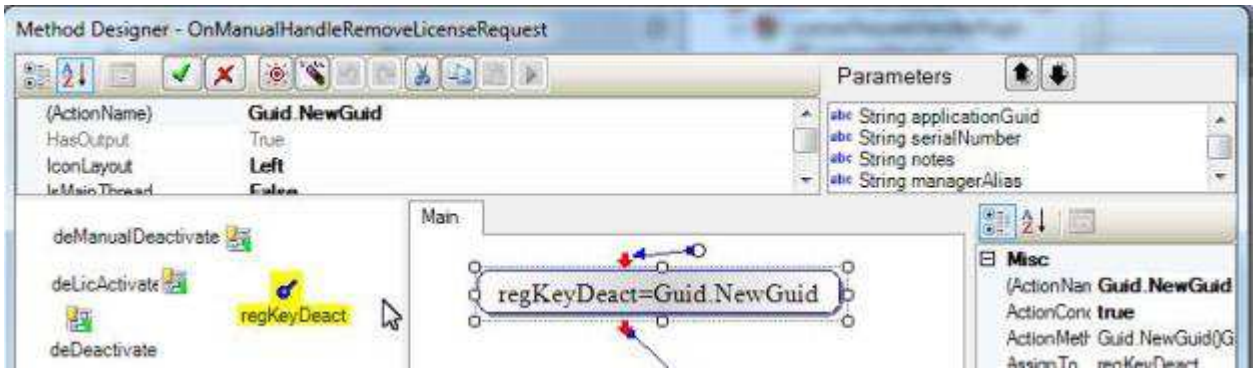






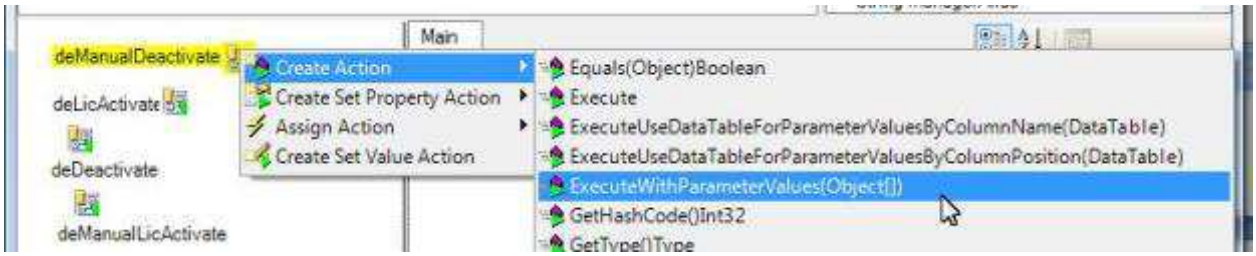


A new key is created:

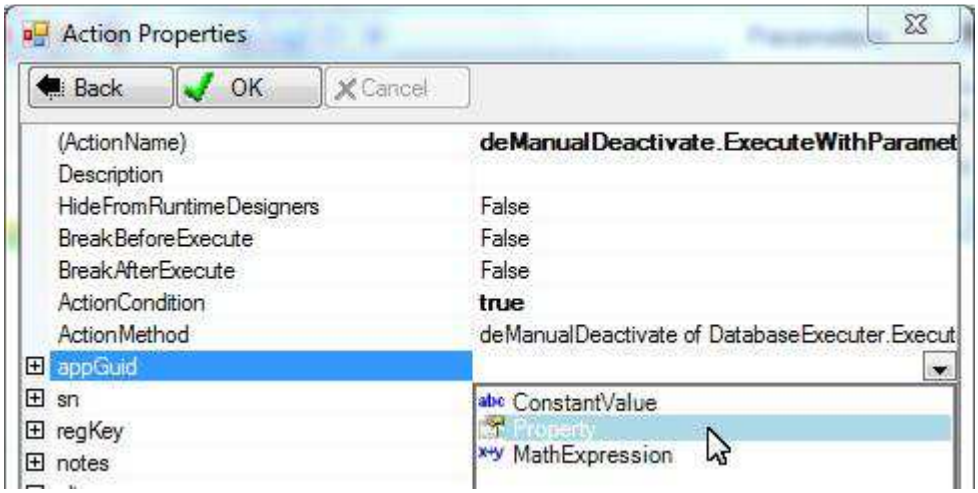


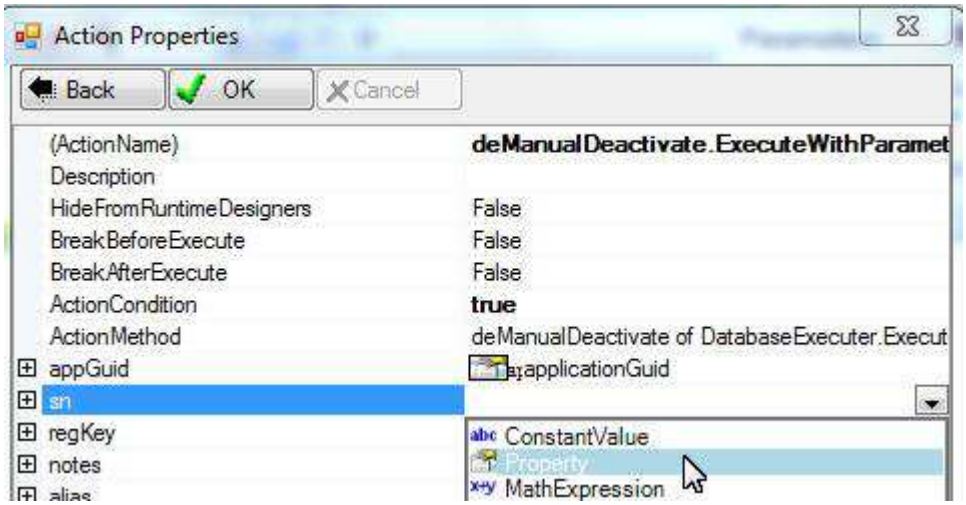
Execute Stored-Procedure

Create an action to execute manual license deactivation stored-procedure:

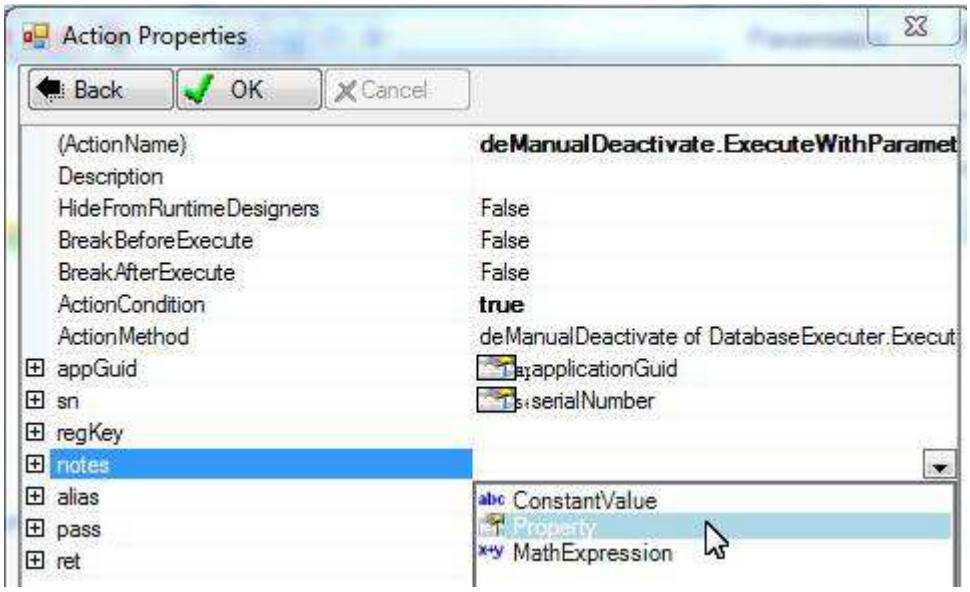


Pass method parameters to the action:

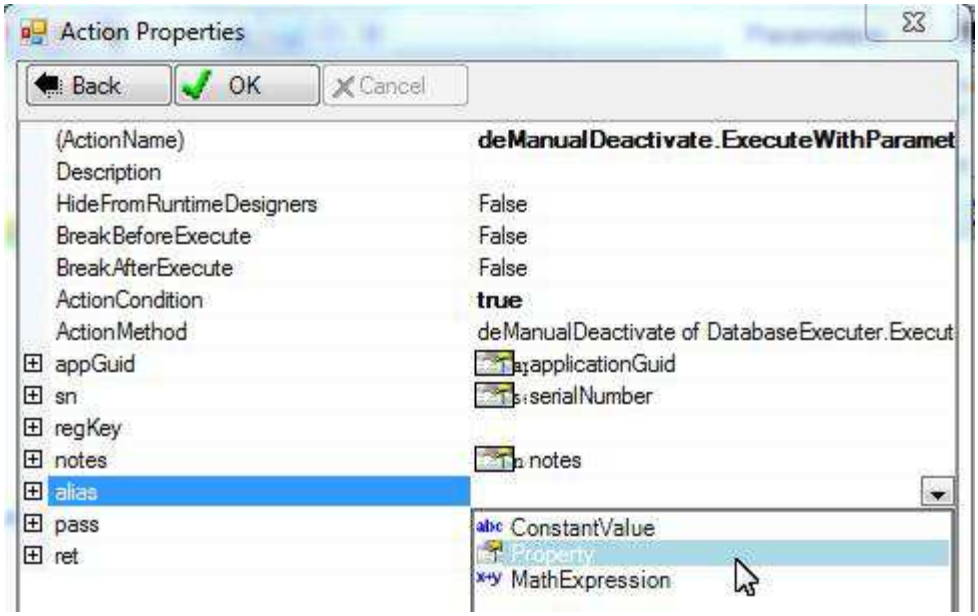


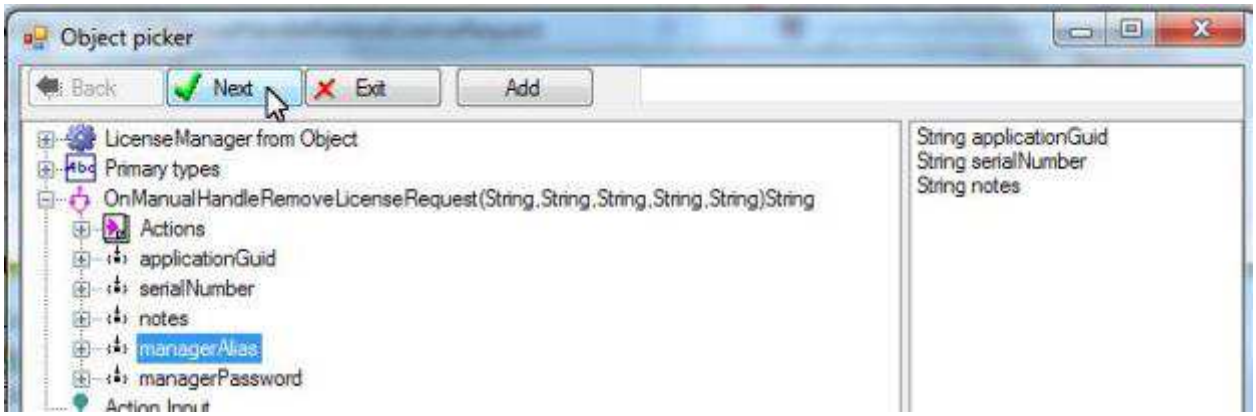

Action Properties
Back
OK
Cancel
(ActionName)
deManualDeactivate.ExecuteWithParamet
Description
HideFromRuntimeDesigners
False
BreakBeforeExecute
False
BreakAfterExecute
False
ActionCondition
true
ActionMethod
deManualDeactivate of DatabaseExecuter.Execut
appGuid
applicationGuid
sn
regKey
notes
ConstantValue
Property
MathExpression

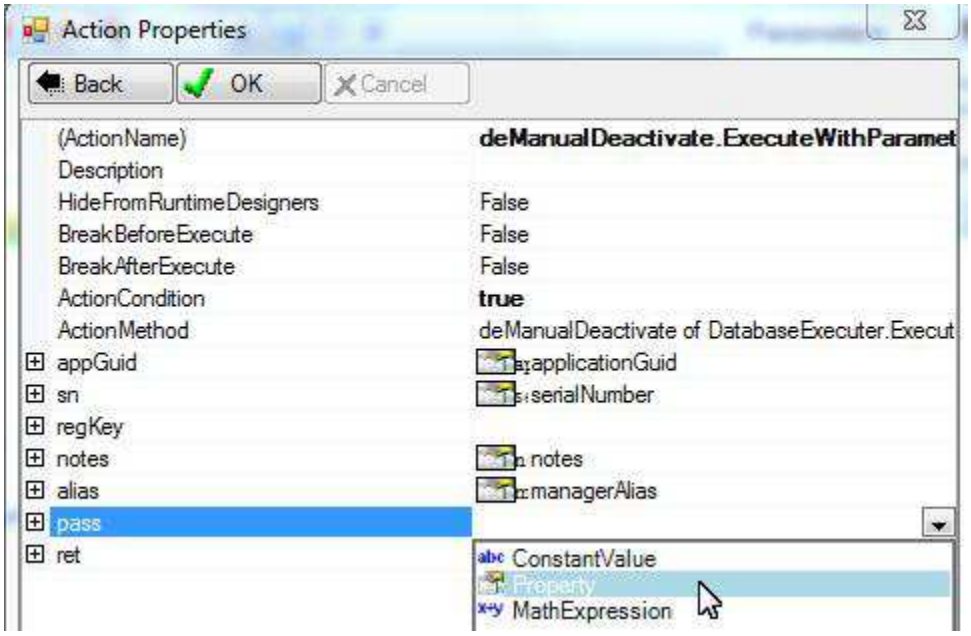

Object picker
Back
Next
Exit
Add
LicenseManager from Object
Primary types
OnManualHandleRemoveLicenseRequest(String,String,String,String,String)String
Actions
applicationGuid
serialNumber
notes
String applicationGuid

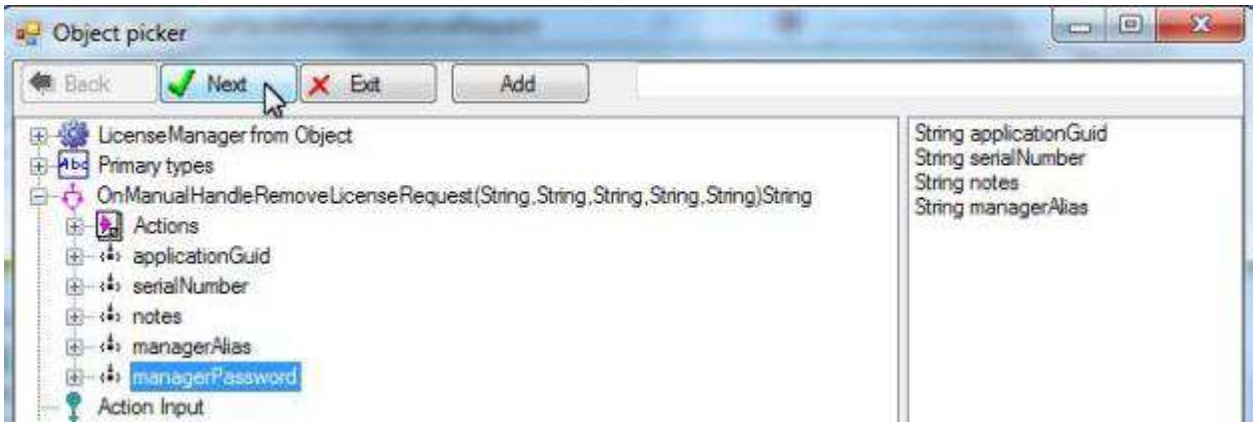

Action Properties
Back
OK
Cancel
(ActionName)
deManualDeactivate.ExecuteWithParamet
Description
HideFromRuntimeDesigners
False
BreakBeforeExecute
False
BreakAfterExecute
False
ActionCondition
true
ActionMethod
deManualDeactivate of DatabaseExecuter.Execut
appGuid
applicationGuid
sn
serialNumber
regKey
notes
alias
pass
ret
ConstantValue
Property
MathExpression


Object picker
Back
Next
Exit
Add
LicenseManager from Object
Primary types
OnManualHandleRemoveLicenseRequest(String,String,String,String,String)String
Actions
applicationGuid
serialNumber
notes
managerAlias
managerPassword
String applicationGuid
String serialNumber

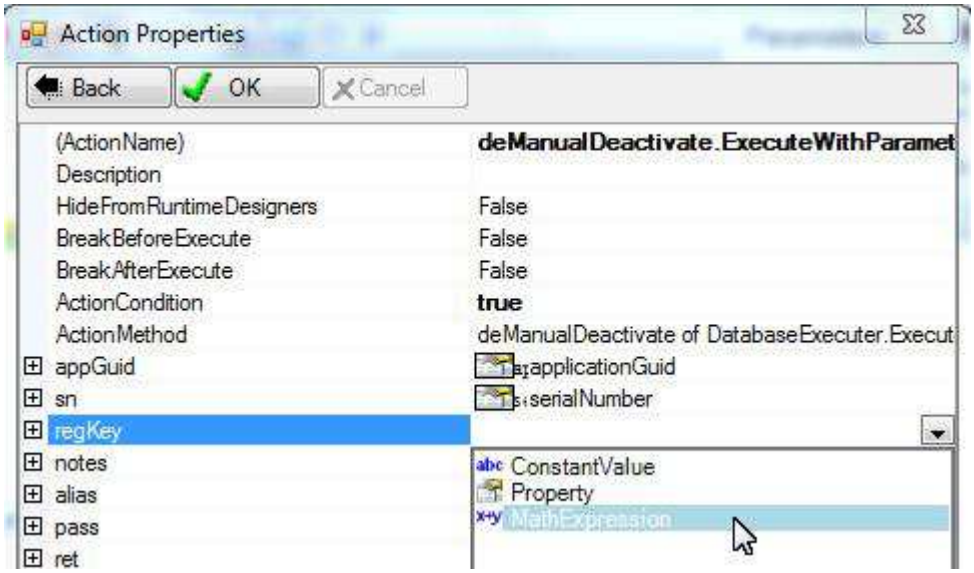

Action Properties
Back
OK
Cancel
(ActionName)
deManualDeactivate.ExecuteWithParamet
Description
HideFromRuntimeDesigners
False
BreakBeforeExecute
False
BreakAfterExecute
False
ActionCondition
true
ActionMethod
deManualDeactivate of DatabaseExecuter.Execut
appGuid
applicationGuid
sn
serialNumber
regKey
notes
notes
alias
pass
ret
ConstantValue
Property
MathExpression

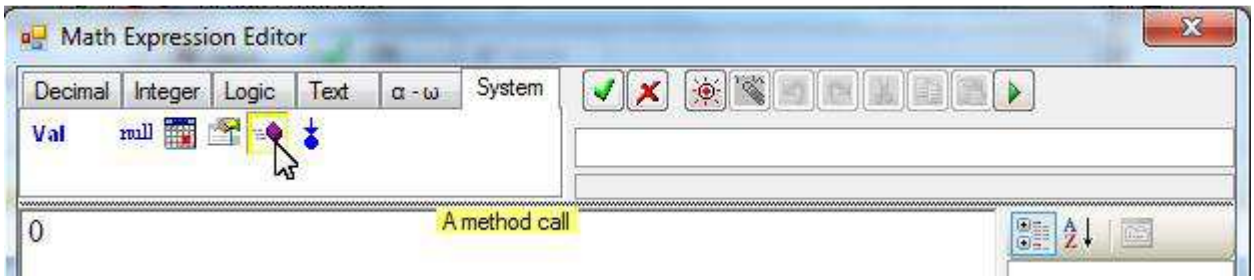

Object picker
Back
Next
Exit
Add
LicenseManager from Object
Primary types
OnManualHandleRemoveLicenseRequest(String,String,String,String,String)String
Actions
applicationGuid
serialNumber
notes
managerAlias
managerPassword
String applicationGuid
String serialNumber
String notes

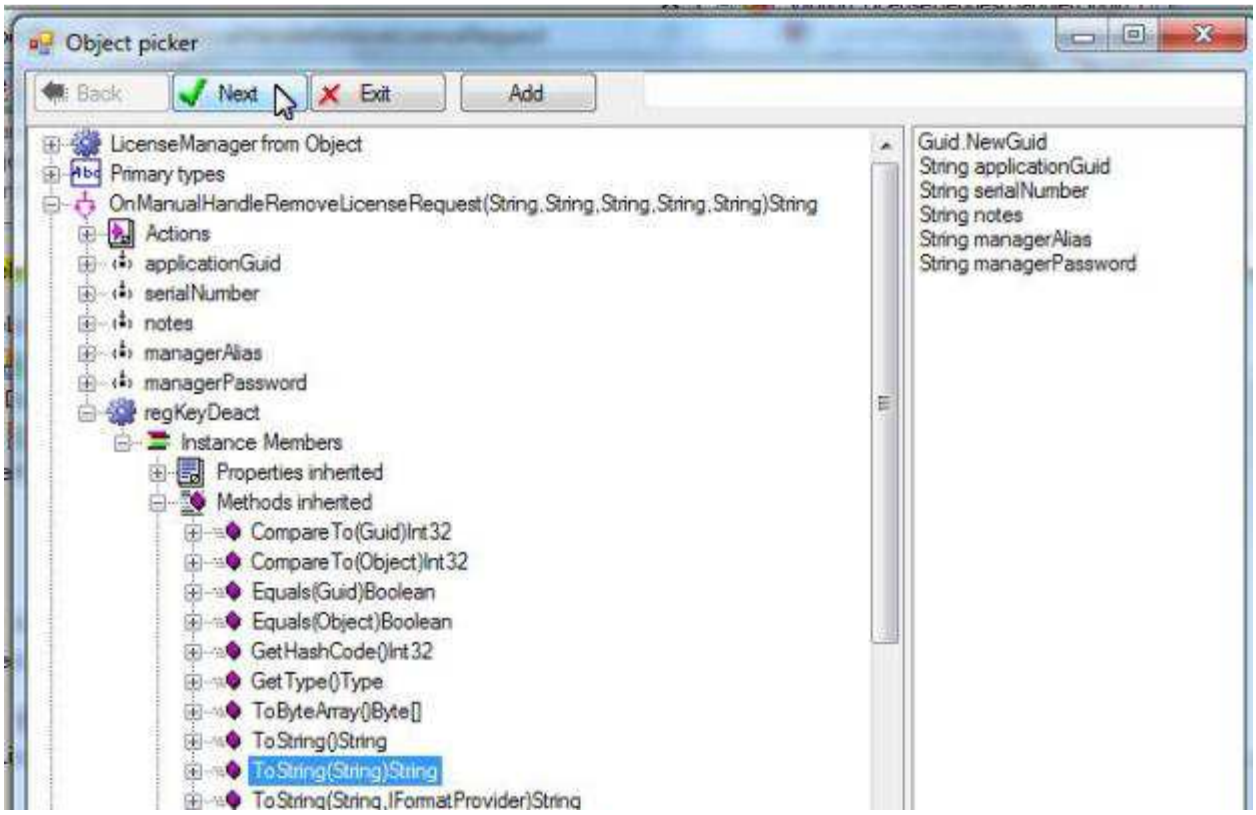


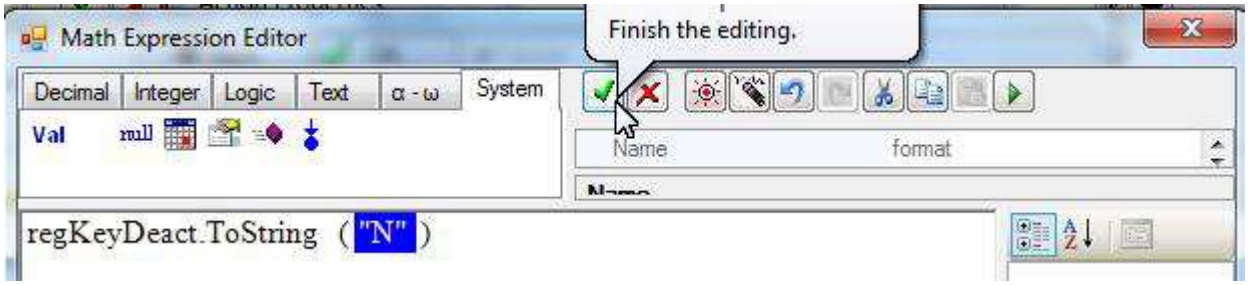


"regKey" is a string. We want to convert the new GUID we created previously to string. So, select "Math Expression":

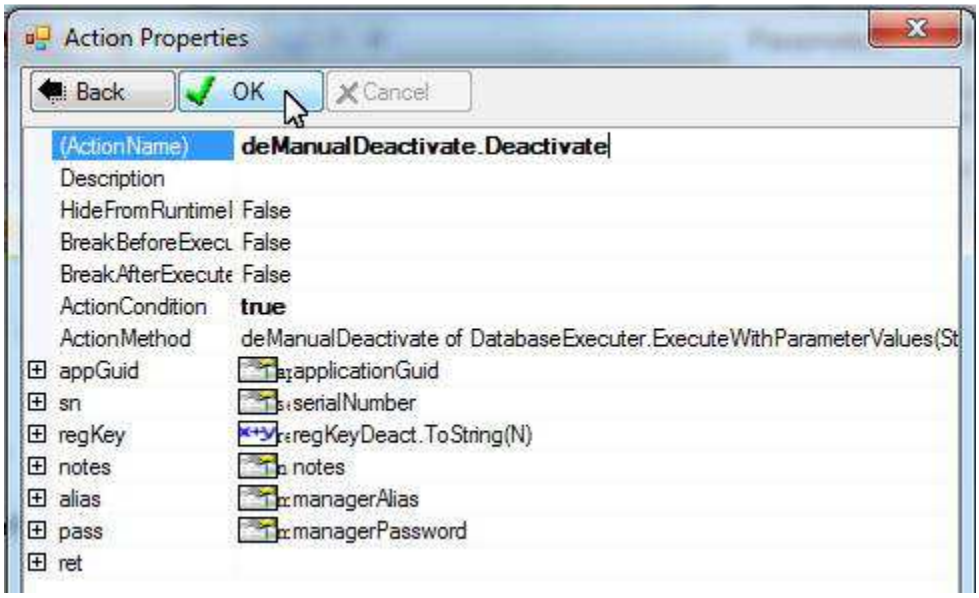

Action Properties
Back
OK
Cancel
(ActionName)
deManualDeactivate.ExecuteWithParamet
Description
HideFromRuntimeDesigners
False
BreakBeforeExecute
False
BreakAfterExecute
False
ActionCondition
true
ActionMethod
deManualDeactivate of DatabaseExecuter.Execut
appGuid
applicationGuid
sn
serialNumber
regKey
notes
alias
pass
ret
ConstantValue
Property
MathExpression


Math Expression Editor
Decimal
Integer
Logic
Text
α - ω
System
Val
mll
A method call
0

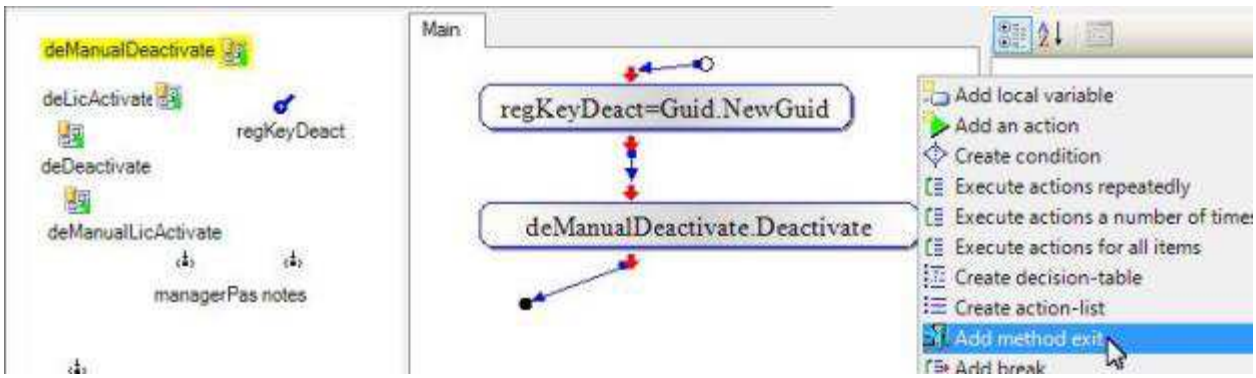

Object picker
Back
Next
Exit
Add
LicenseManager from Object
Primary types
OnManualHandleRemoveLicenseRequest(String,String,String,String,String)String
Actions
applicationGuid
serialNumber
notes
managerAlias
managerPassword
regKeyDeact
Instance Members
Properties inherited
Methods inherited
CompareTo(Guid)Int32
CompareTo(Object)Int32
Equals(Guid)Boolean
Equals(Object)Boolean
GetHashCode()Int32
GetType()Type
ToByteArray()Byte[]
ToString()String
ToString(String)String
ToString(String,IFormatProvider)String
Guid.NewGuid
String applicationGuid
String serialNumber
String notes
String managerAlias
String managerPassword

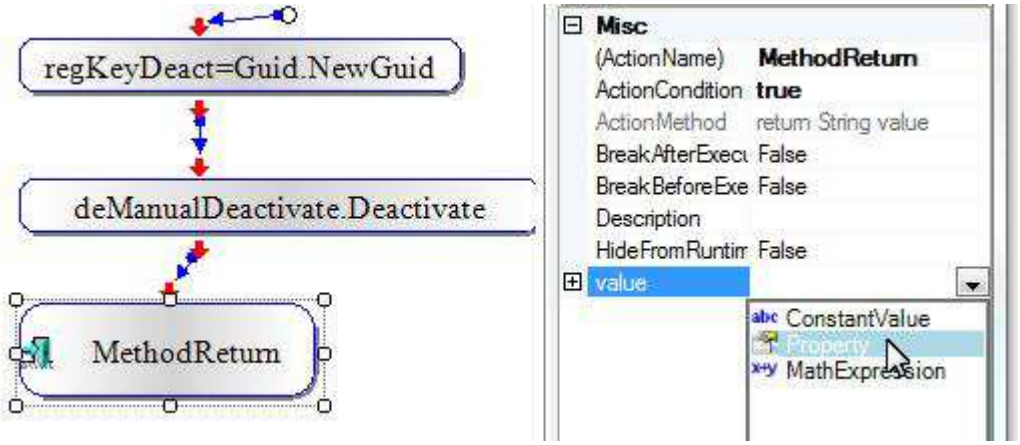


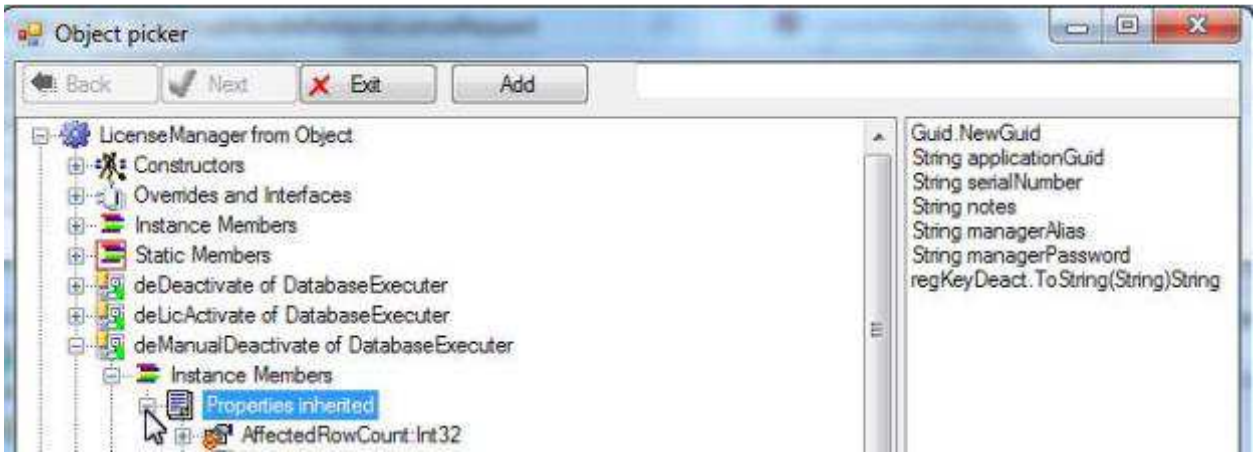
Rename the action and click OK:

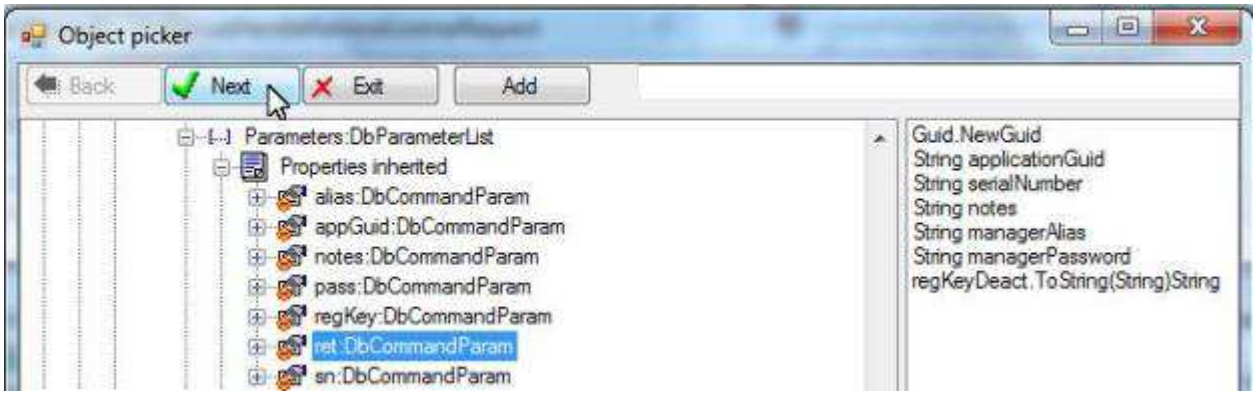


Create a method return action to return parameter “ret” of the DatabaseExecuter:

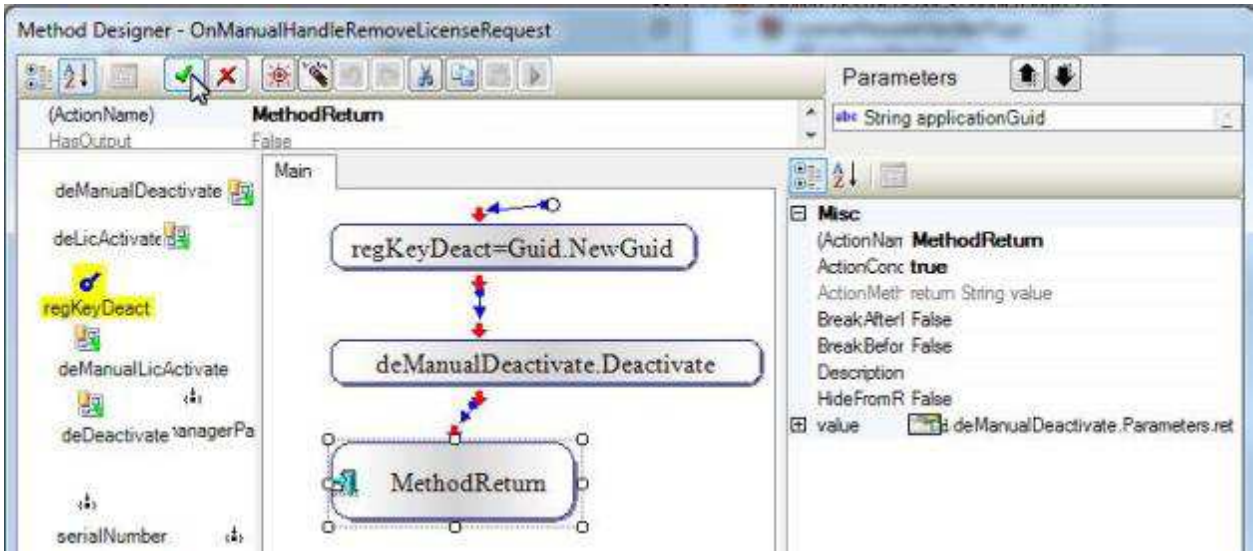






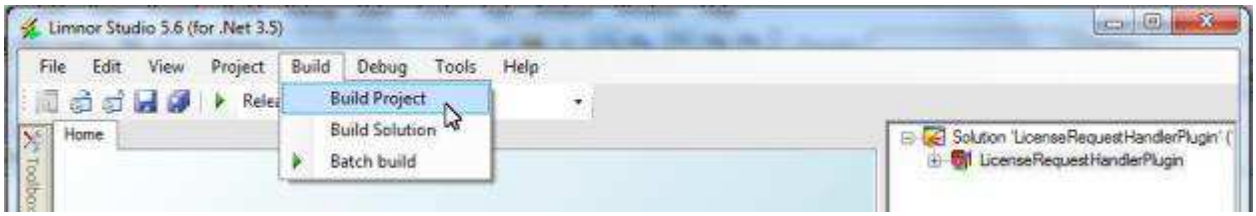


We are done modifying this method.

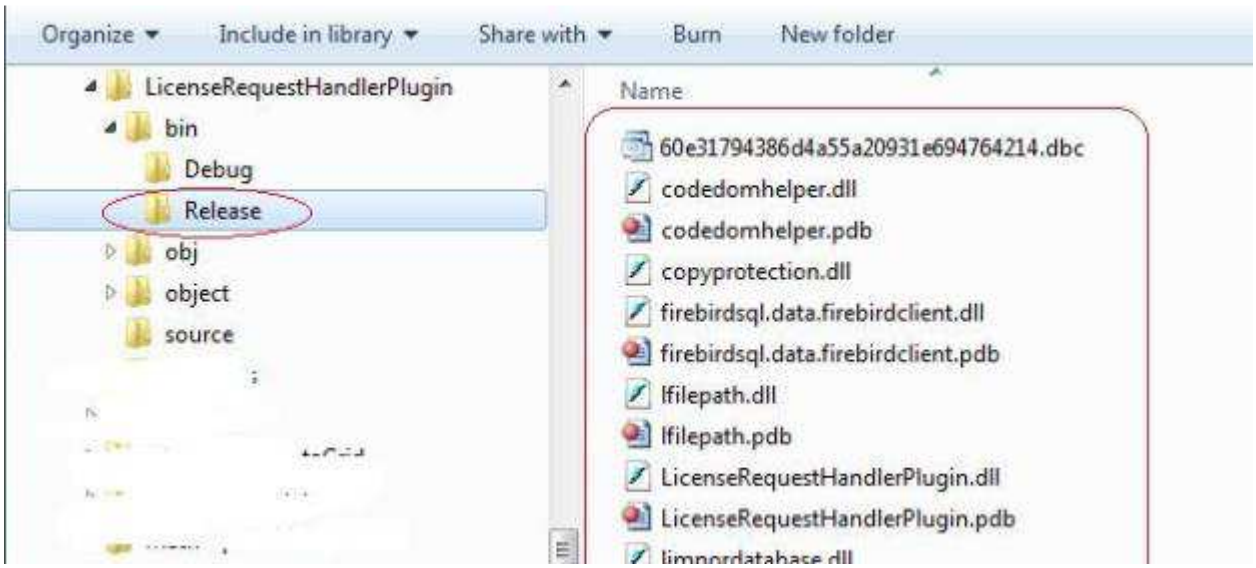


Deploy License Management Plug-in

Compile License management plug-in project:



All files for the license management plug-in can be found under bin\release folder of the project folder:



Copy all the files to the **bin** folder of the web application.

Test License Management Plug-in

Create Application Record

We need to record all copy-protected applications in our database. For our sample application, we may use following script to record it in the database:

```
INSERT INTO softwareProtected (AppGuid, AppName, CreateTime)
 VALUES ('e1842b90029e466ead5724e69e64ec6c', 'CopyProtectedAppSample', (SELECT NOW()));
```

Run it in MySQL console:

Create Serial Number Records

When you issue a license you'll record following information to the database:

- Serial Number – It is a string no more than 60 characters long. It should be unique for one application.
- License Code – It is an integer. It can be 1, 2, 3, or 4.

License Code	Meaning
1	No expiration. Hardware signature not checked
2	Use expiration. Hardware signature not checked
3	No expiration. Hardware signature is checked
4	Use expiration. Hardware signature is checked

- Feature Code – It is an integer representing features allowed for this license.
- Expire Time – It is a date. If License Code is 2 or 4 then an expiration time should be given.

The MySQL script includes a stored-procedure for easier of recording serial number:

```
CREATE PROCEDURE AddSerialNumber (IN appGuid VARCHAR(40), IN sn VARCHAR(60), IN licenseCode INT,
IN features INT, IN expireTime DATETIME)
```

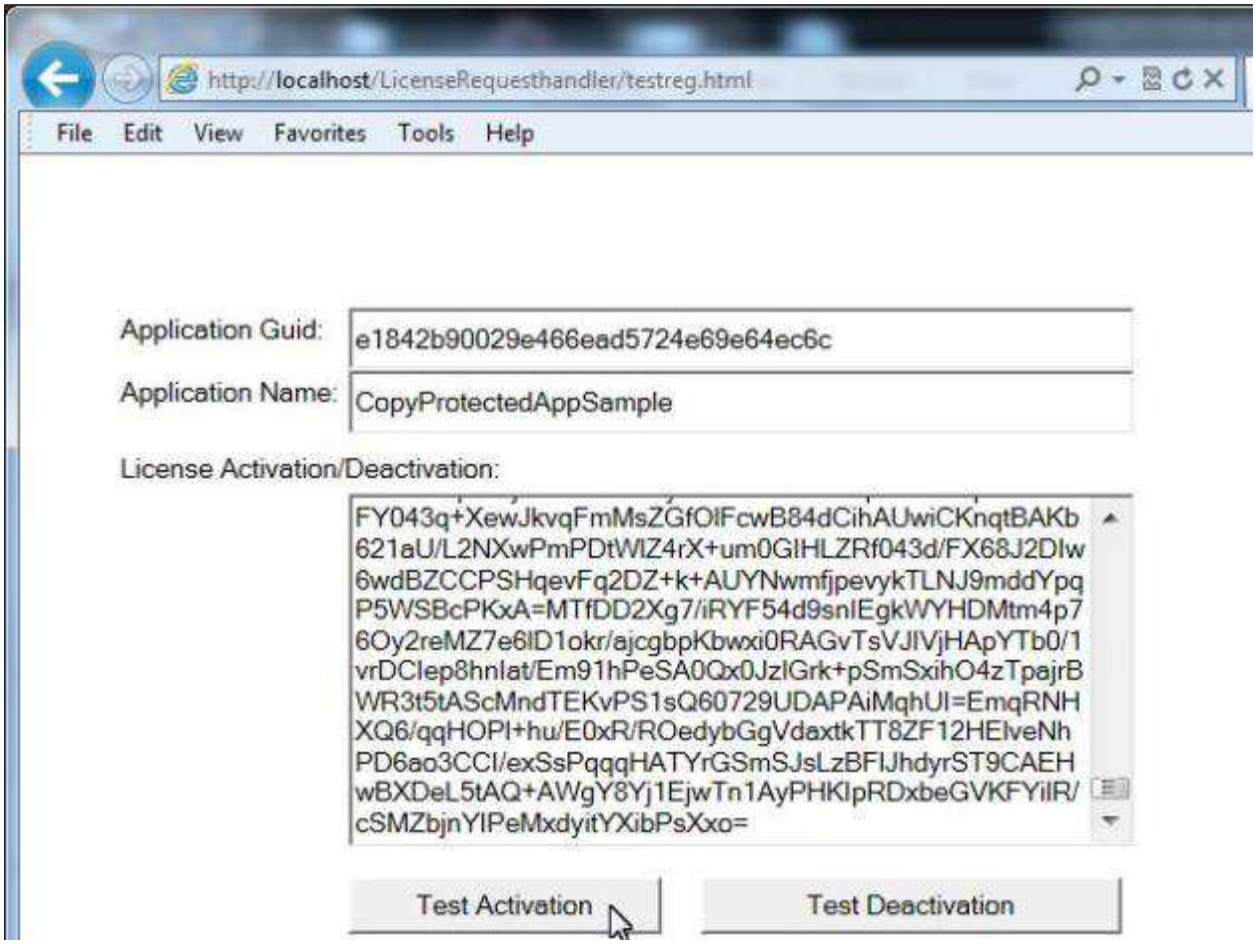
We may call it to add serial numbers to our database.

For example, suppose we want to add a serial number "12345678" with License Code 1 and Features 6, indicating that it is an unlimited license and allowing the 2nd and 3rd features:

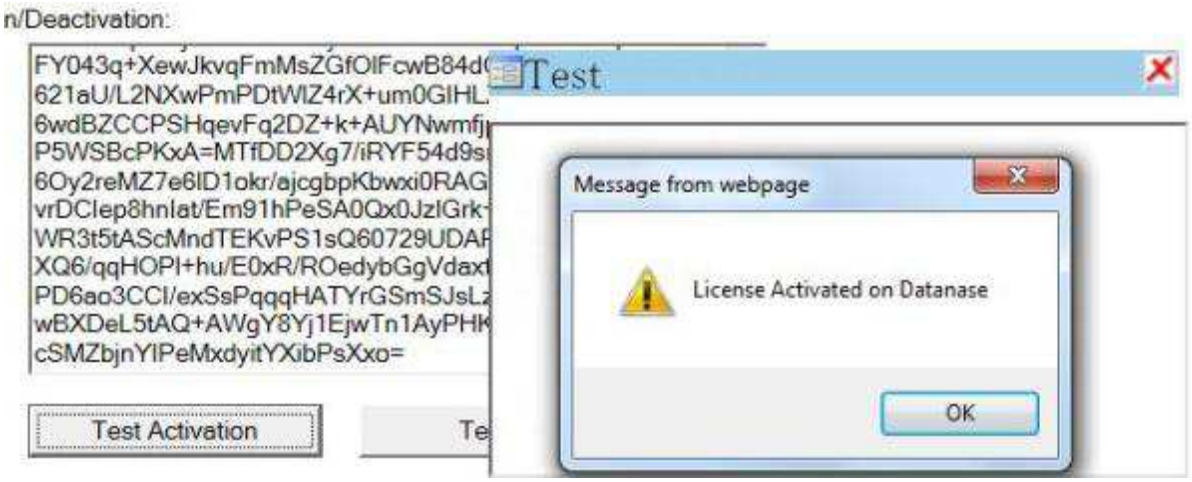
Test web application

Now we may re-test our web application deployment.

Launch testReg.html from a web browser. Copy contents of an activation file to the text box on the test web page. Click "Test Activation".



Wait for some time. A message box appears:

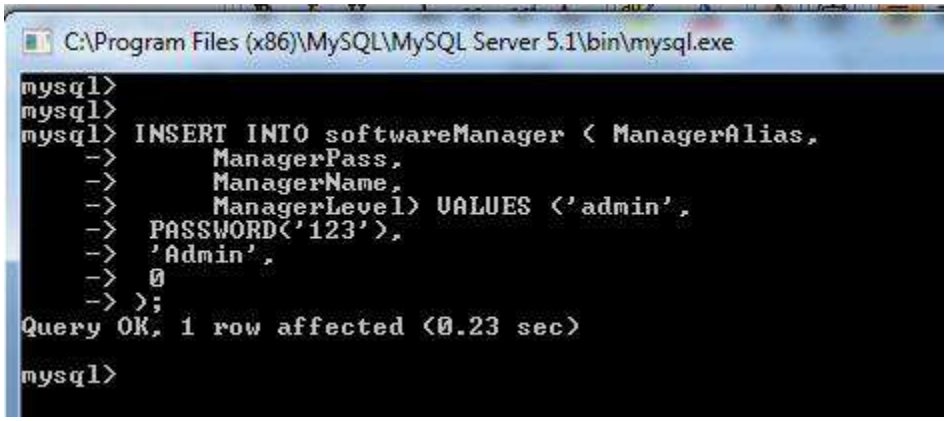


If you reached this stage then the deployment of web application and license management plug-in is done.

We may start some operation testing.

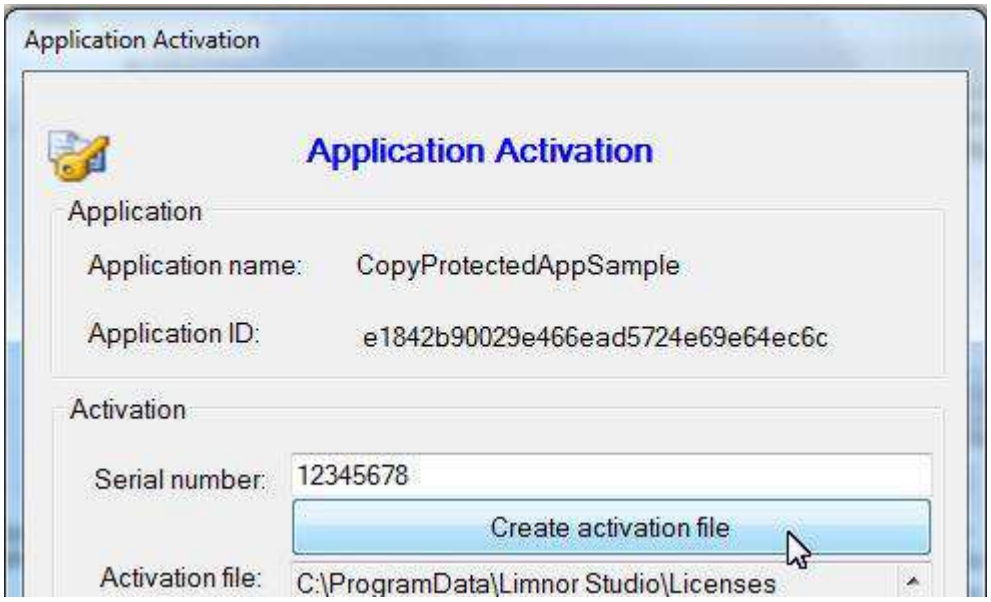
Add Software Manager Accounts

For unconditionally deactivate a license we need to have an account in the database so that we have permission to do such operations.

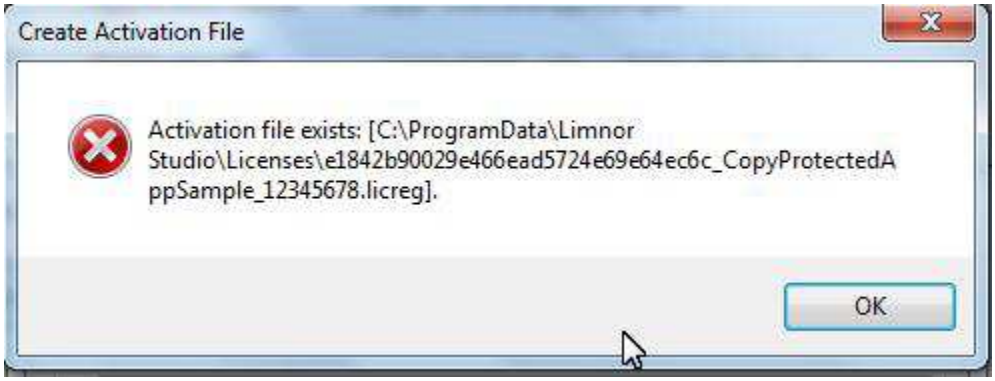


Deactivate License Unconditionally

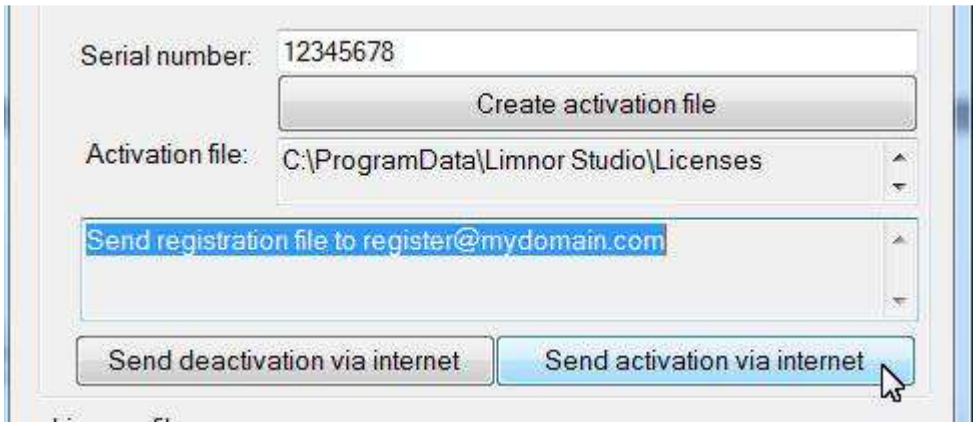
A test by TestReg.html actually damaged the automated license activation/deactivation. To show this damage, let's run our copy-protected application. It asks for an activation file:

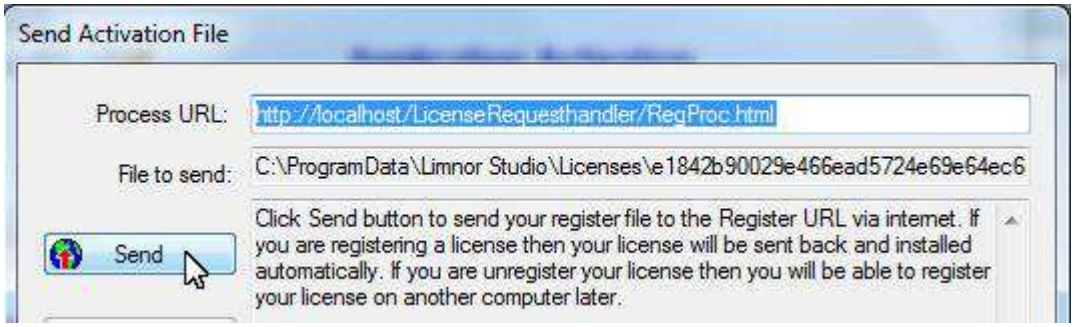


But it does not allow us to create a new activation file because a file exists:



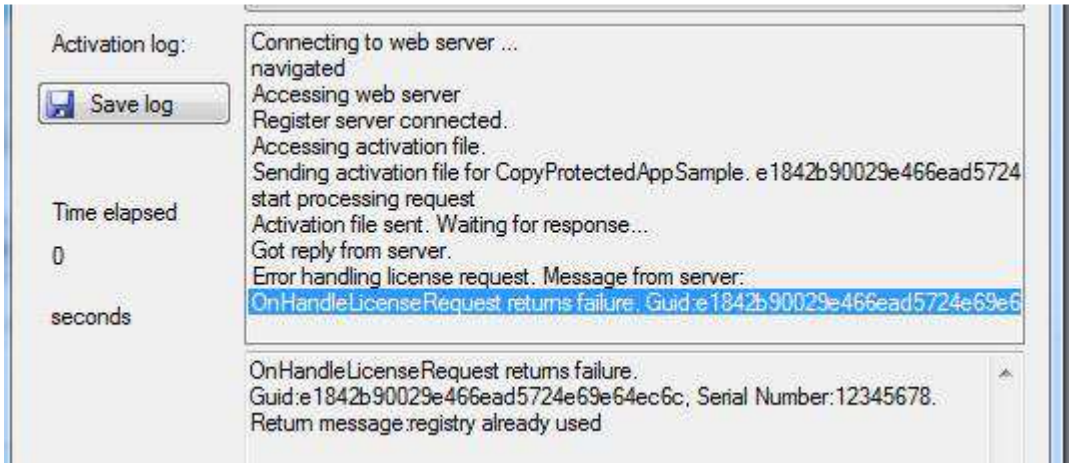
If we try to send the existing activation file then it will also fail:



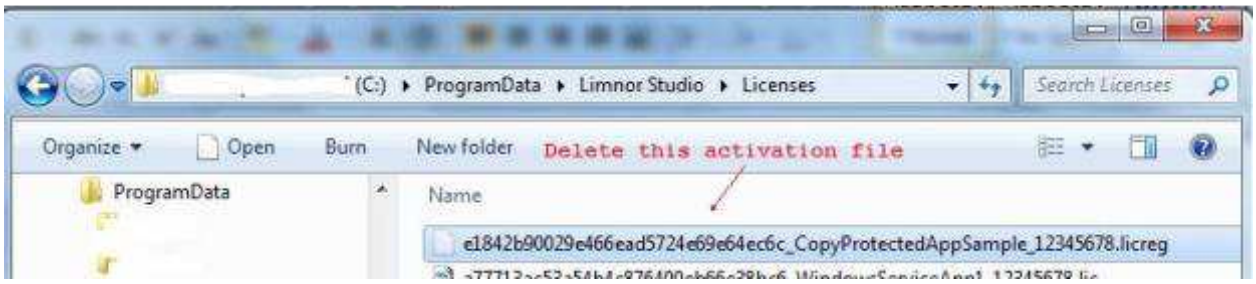


(**Warning**: web browser window for testReg.html must be closed before you click "Send". If you click Send and it takes long time without result then that is the cause of the problem.)

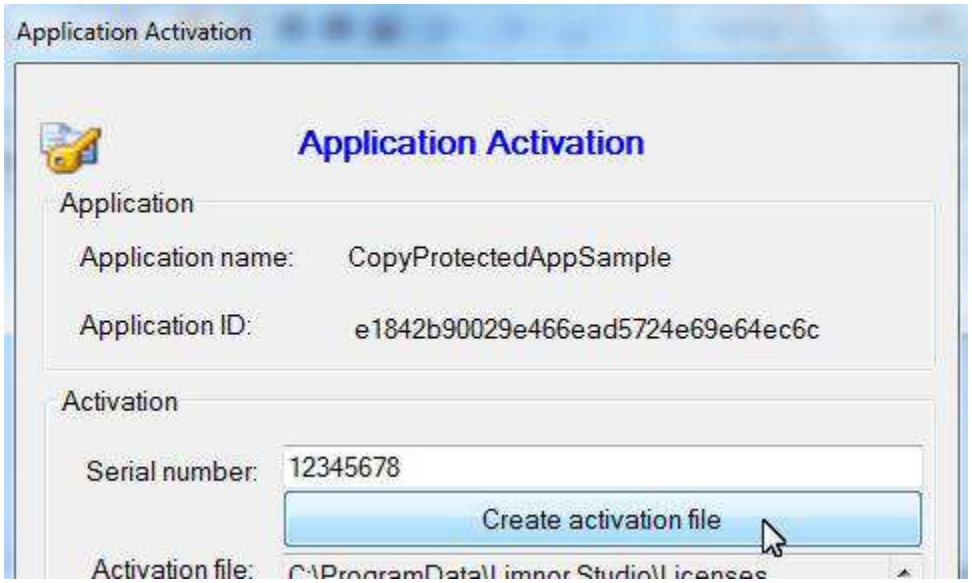
We get an error message that the registry is already used:



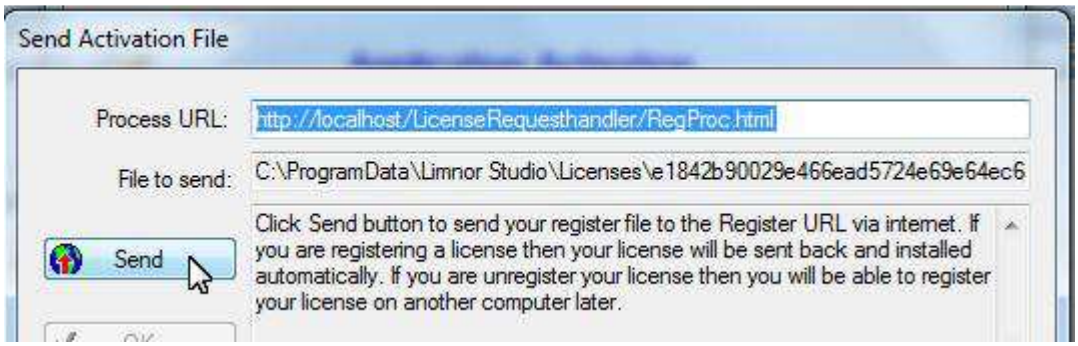
We get this result because we used the registry key on testReg.html. We may delete the activation file so that we may create a new activation file.



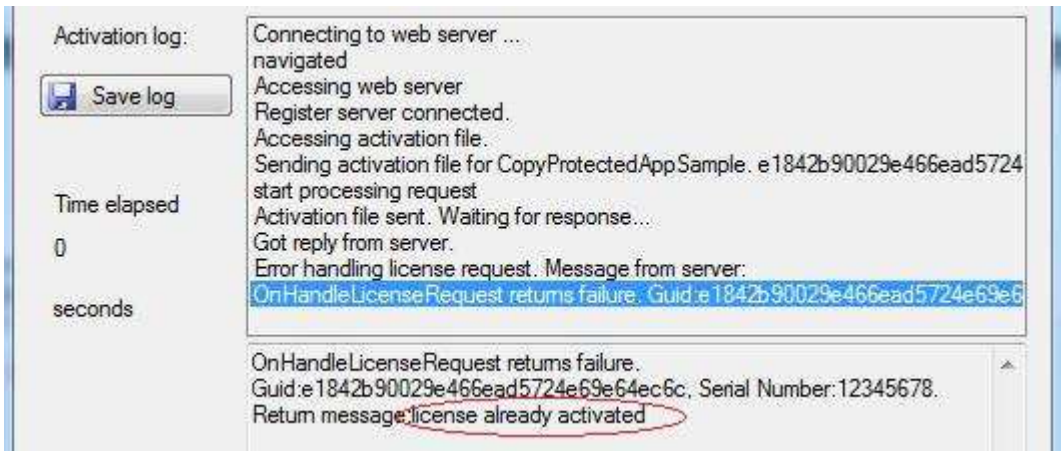
Now click "Create Activation File":



File sending dialogue box appears:

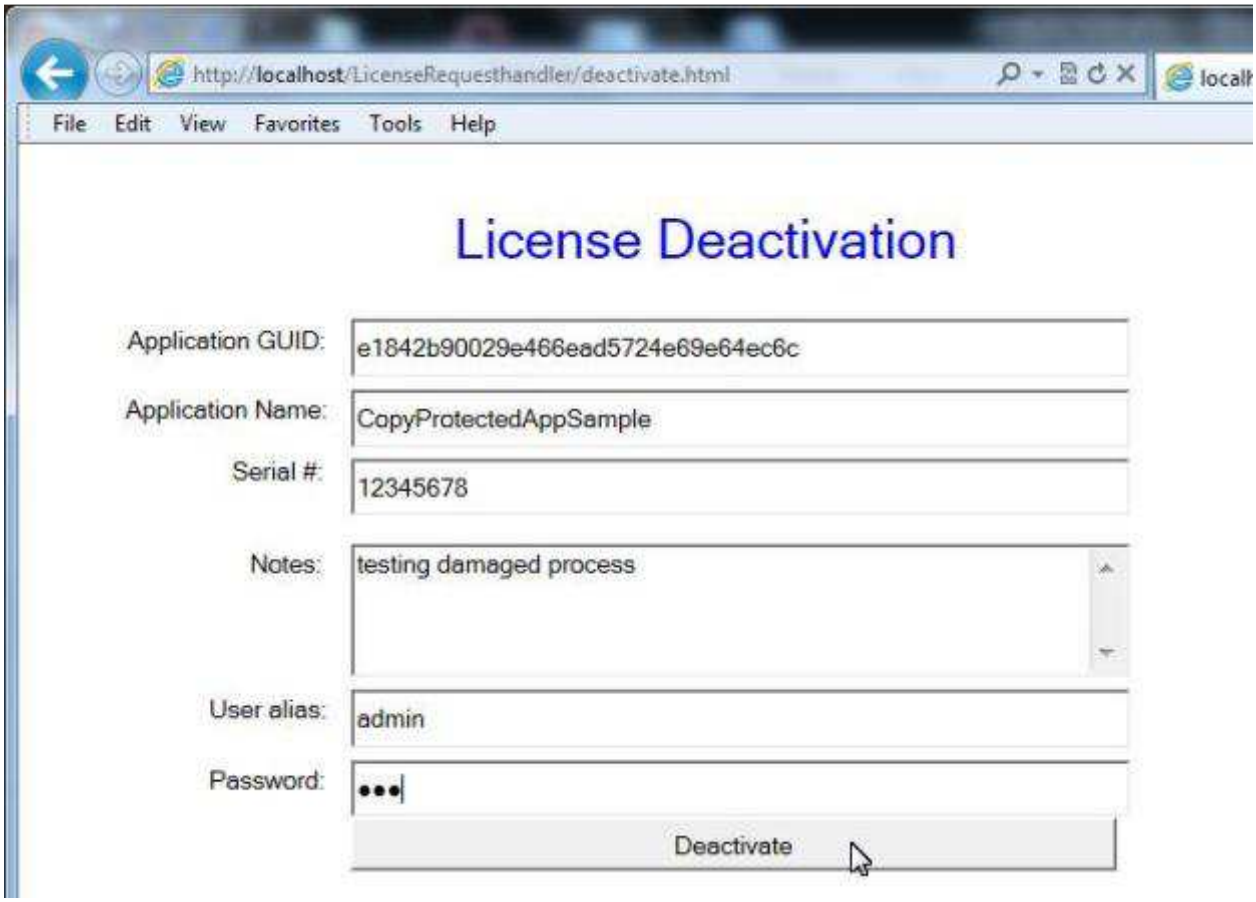


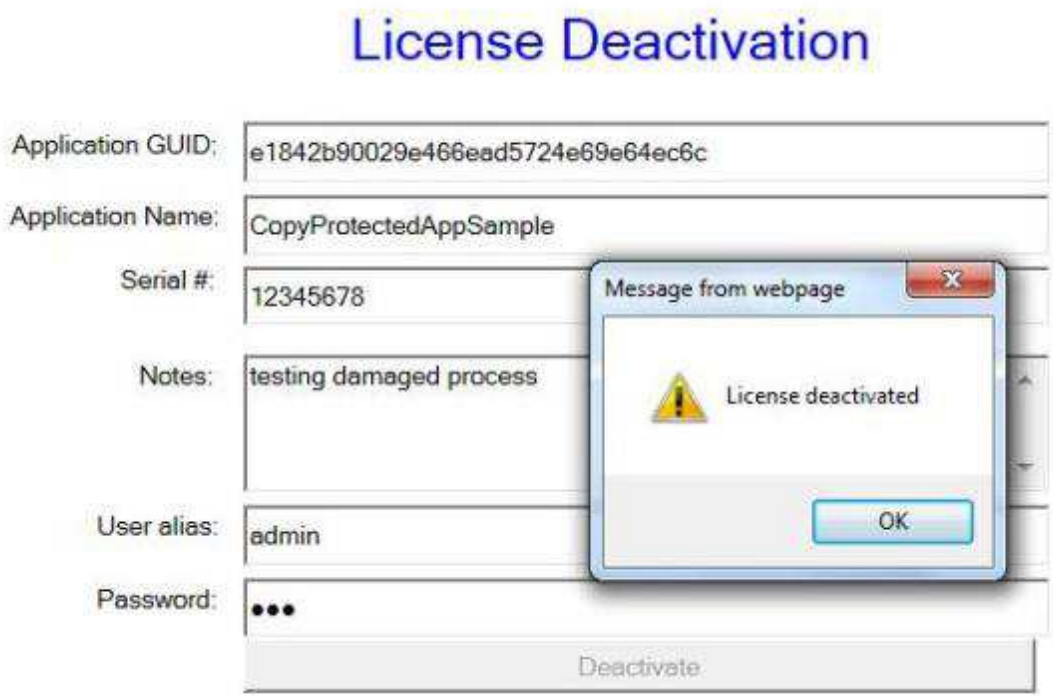
This time the result is “license already activated”:



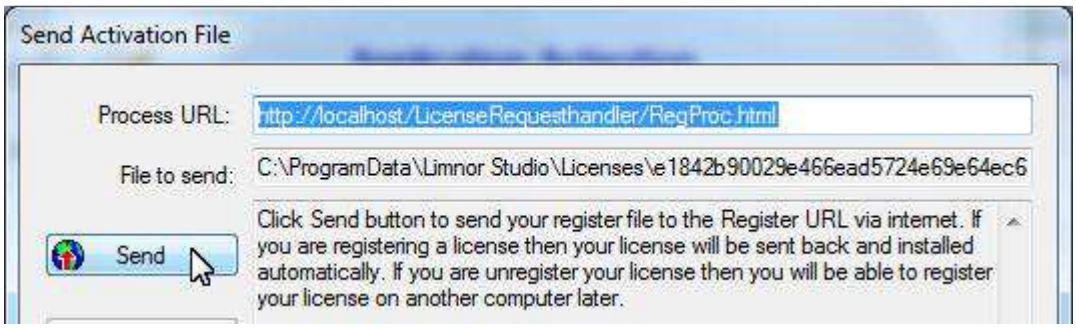
Now we are going to use Deactivate.html to unconditionally deactivate the license so that our new activation file may go though.

Launch Deactivate.html in a web browser. Enter application information and credential. Click "Deactivate":

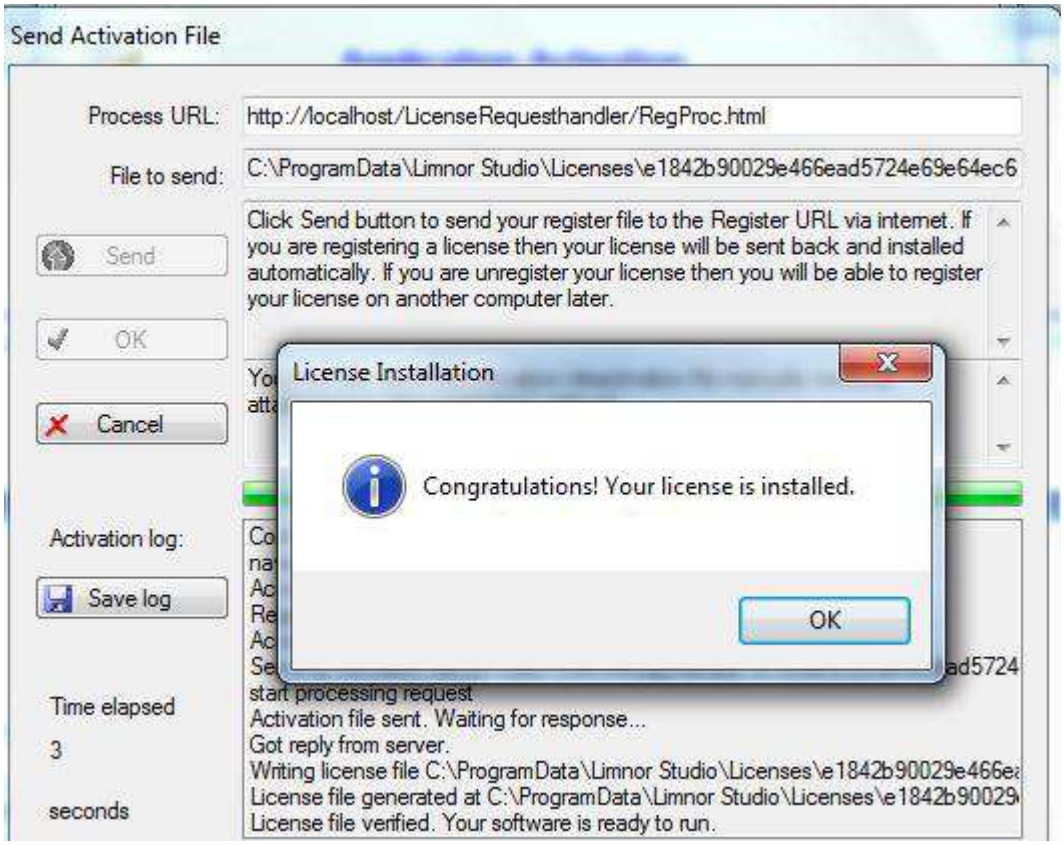




Now let's send the activation file again:

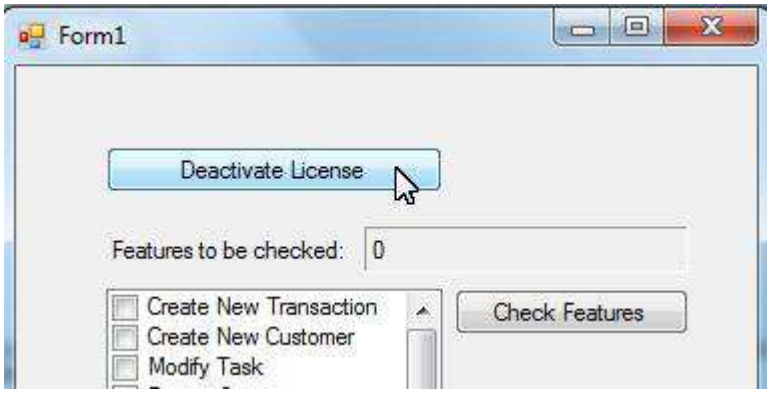


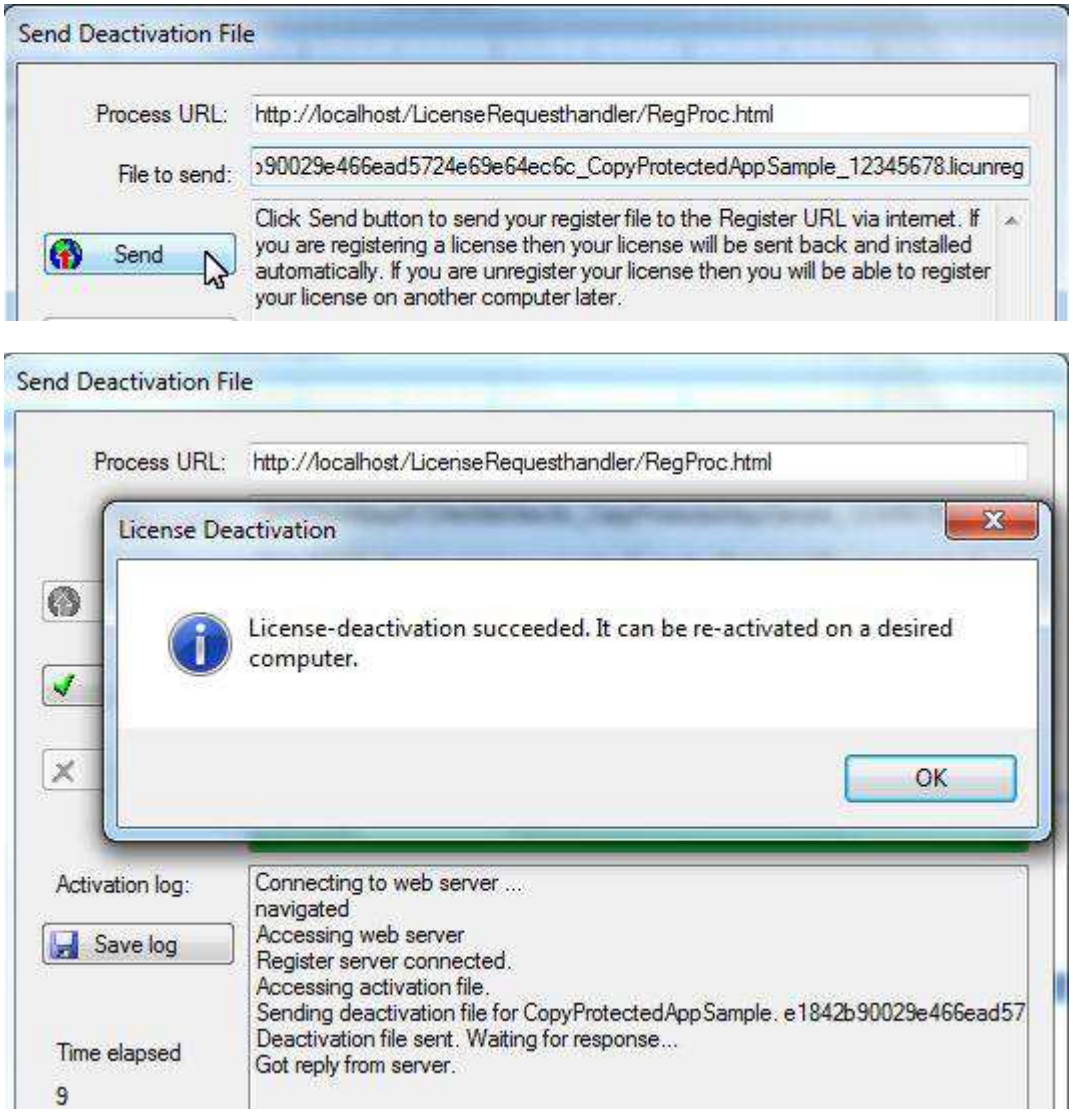
This time the activation file goes through:



Deactivate License Automatically

Once a license is activated, the application runs normally. If you give your application a chance to deactivate the license then deactivation information may also be sent to your web server.

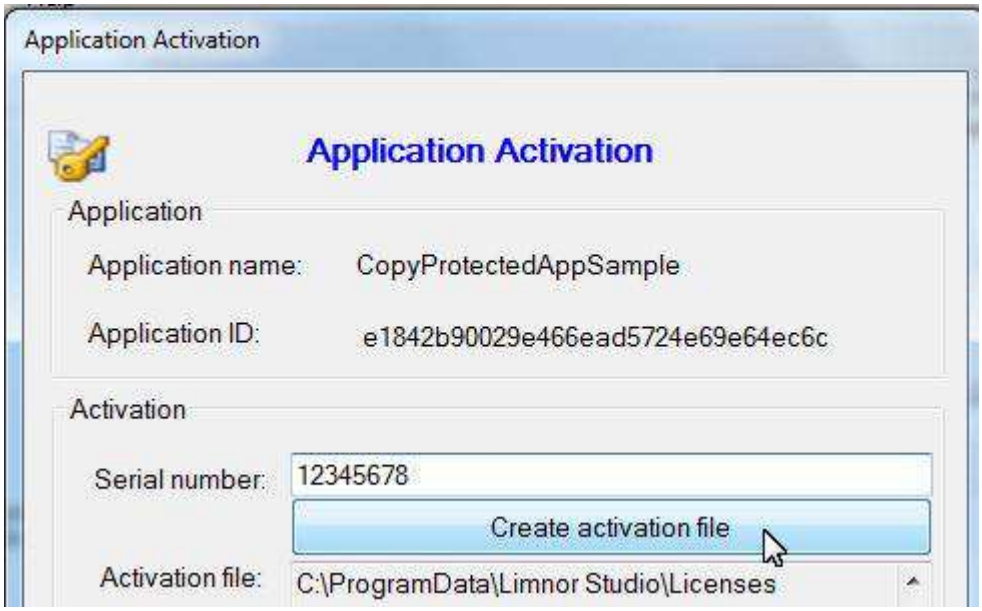




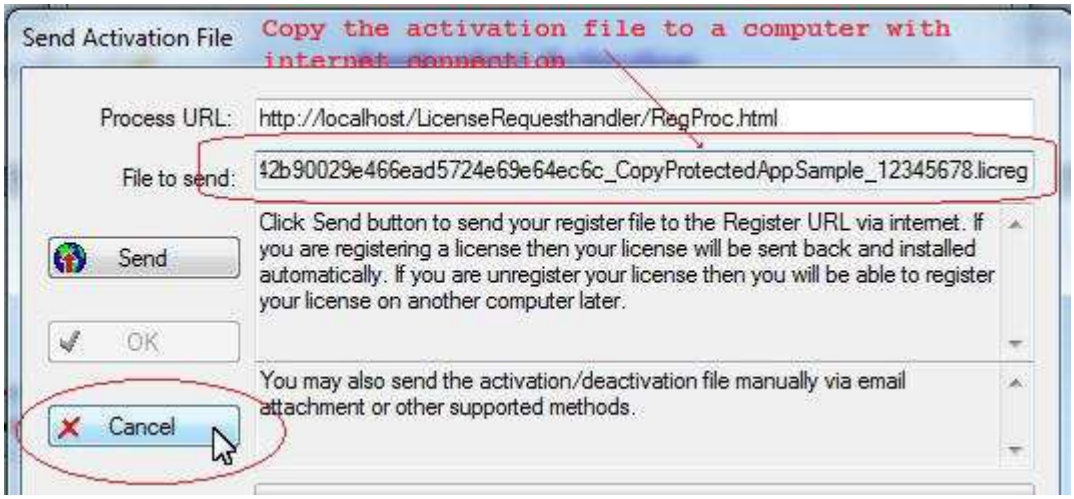
Send Activation File Manually

If a user's computer does not have internet connection then it cannot use automated license activation and deactivation. The user may access your web page, **ActivationManager.html**, from another computer having internet connection to manually send activation and deactivation files.

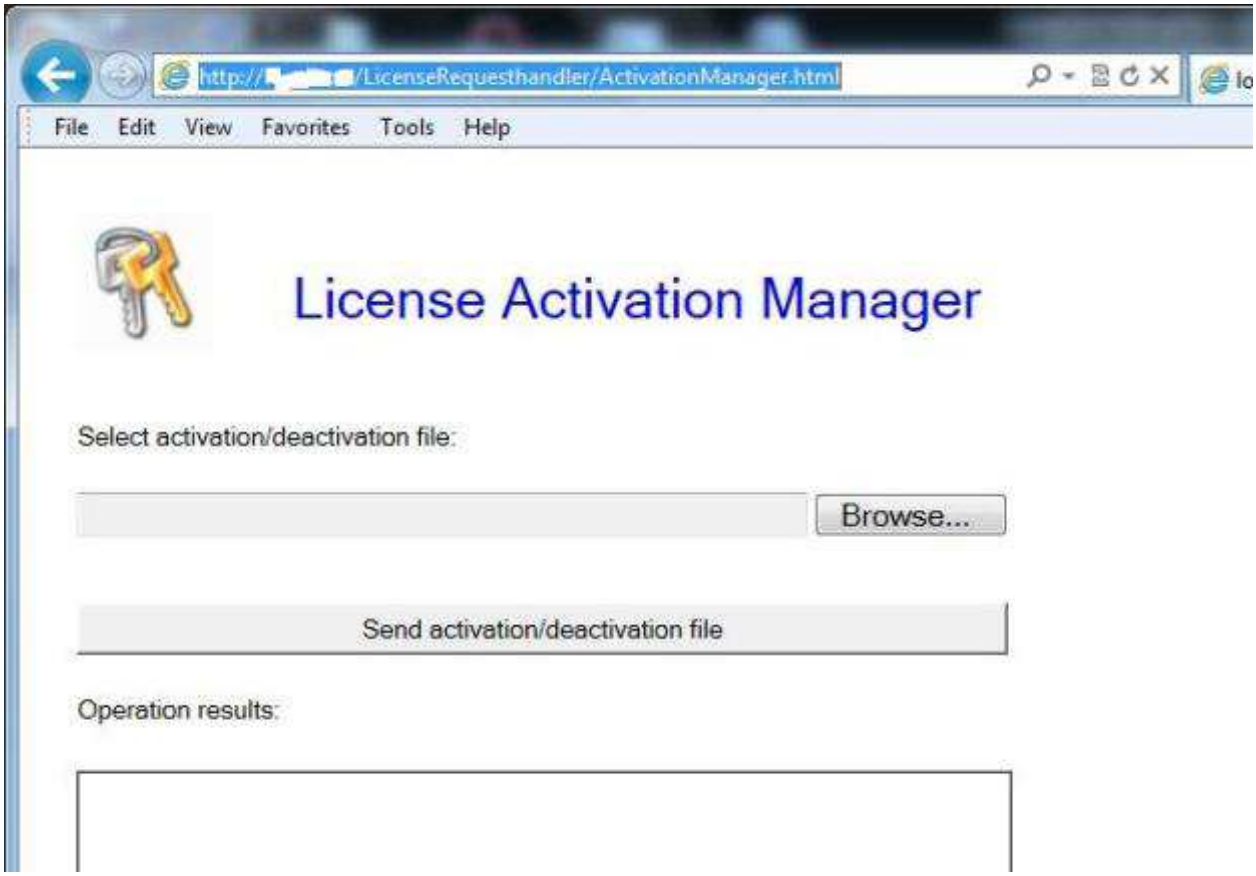
Run the sample application, create an activation file:



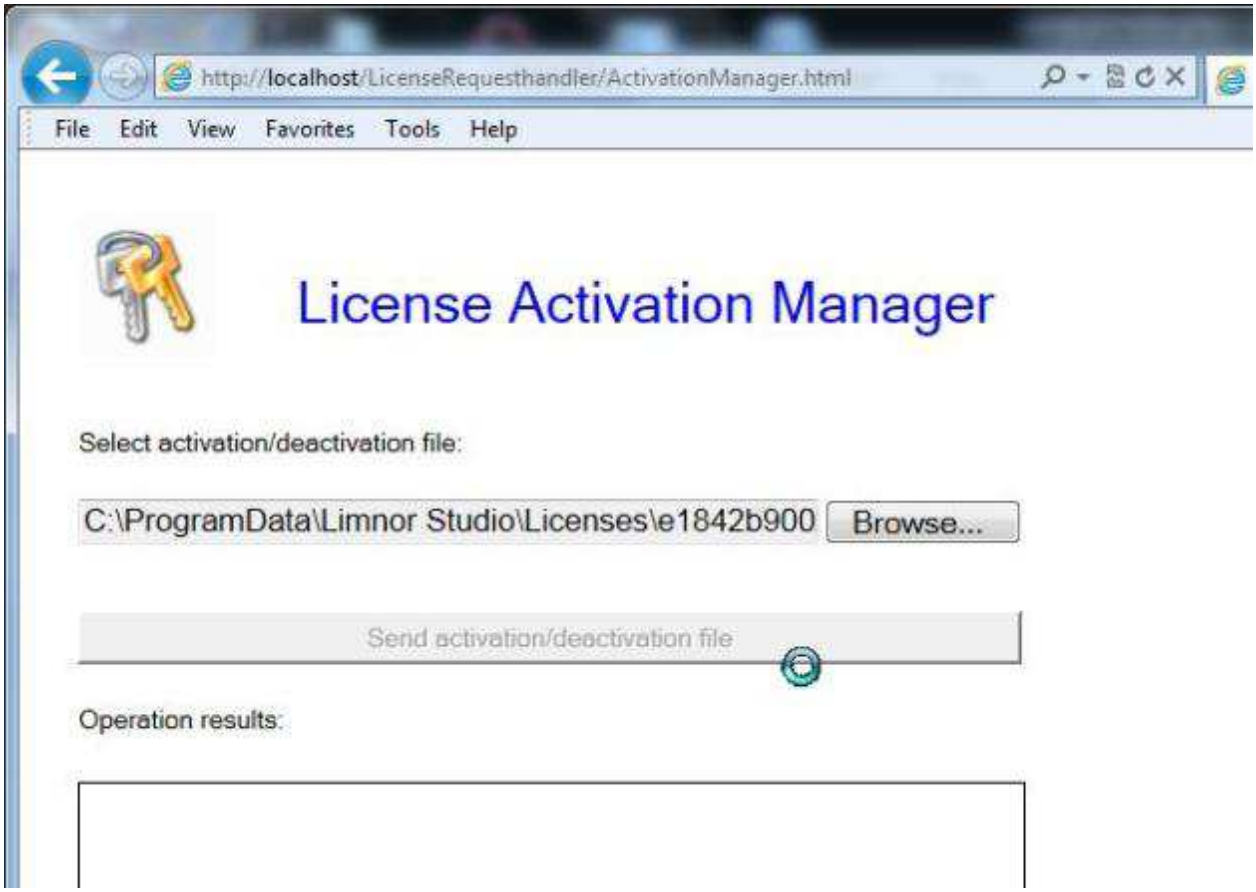
The file sending dialogue box appears. Copy the activation file to a computer which has internet connection. Click Cancel button to close the dialogue box.



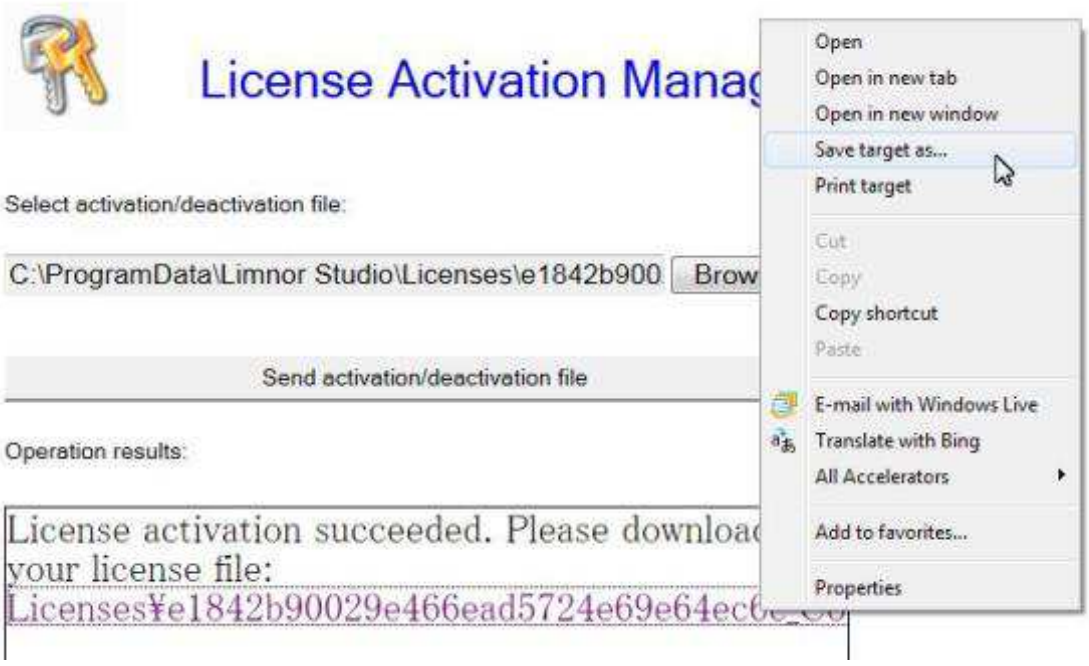
From the computer with internet connection, your user opens your web page ActivationManager.html:



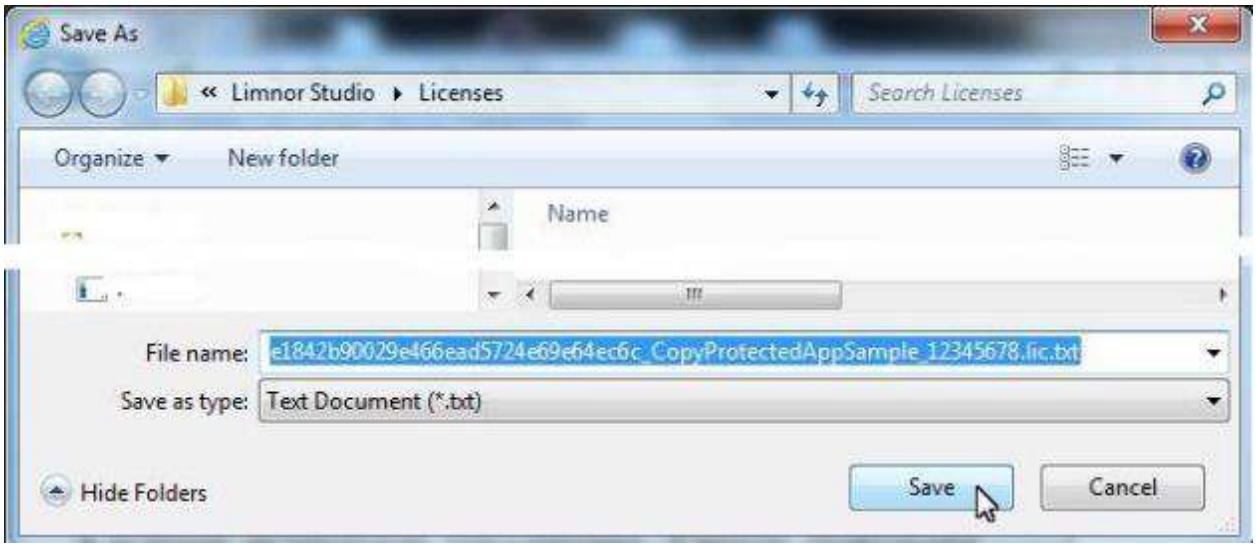
Your user locates the activation file and click “Send activation/deactivation file”:



The activation file is processed on the web server and a license file is generated for your user to download. Right-click a download URL and choose “Save target as…”:

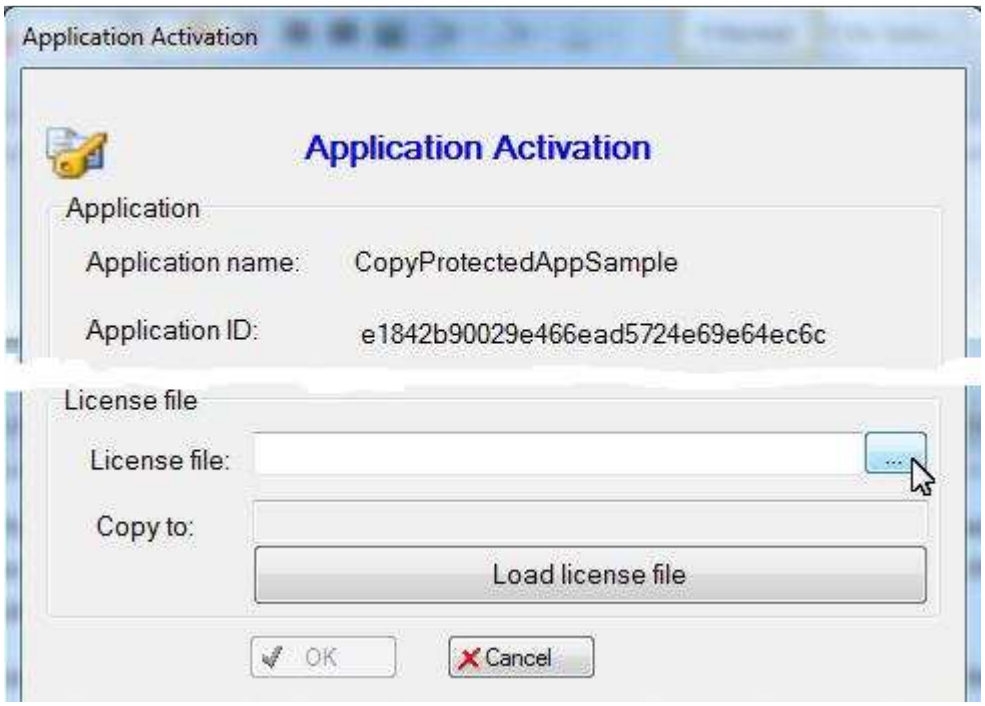


Choose a folder to save the license file:

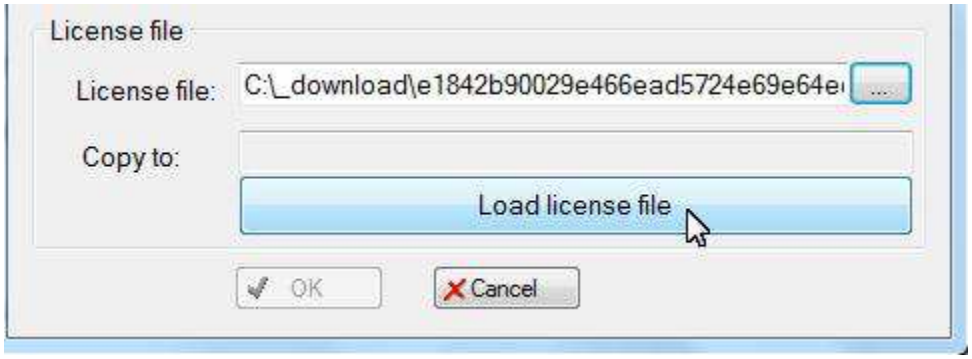


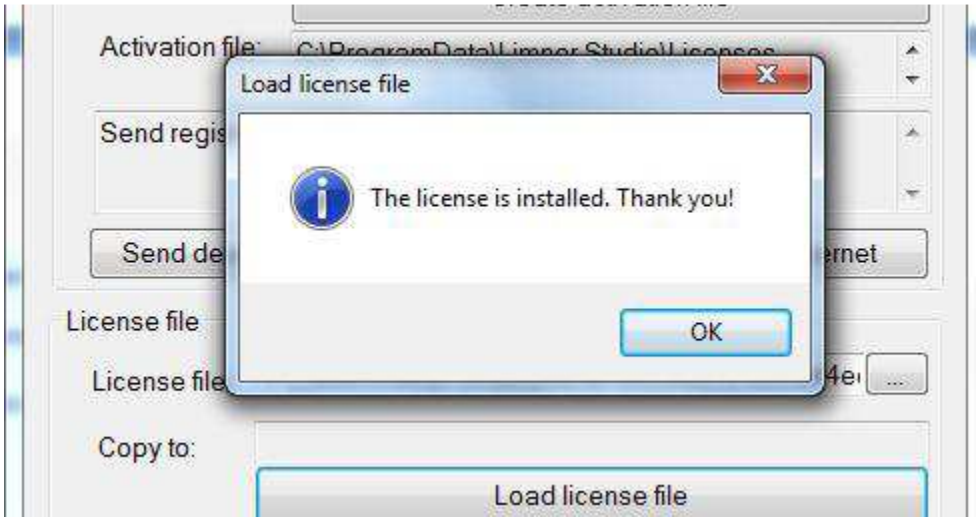
Note that the license file downloaded uses file extension ".txt". This is because that many web servers do not allow downloading files with file extension ".lic". Once your user downloads the file, the file name should be changed by removing ".txt".

Your user copies downloaded and renamed license file to the computer where copy-protected software is installed. Run the software, it asks for license file. Click "…" to find downloaded file.

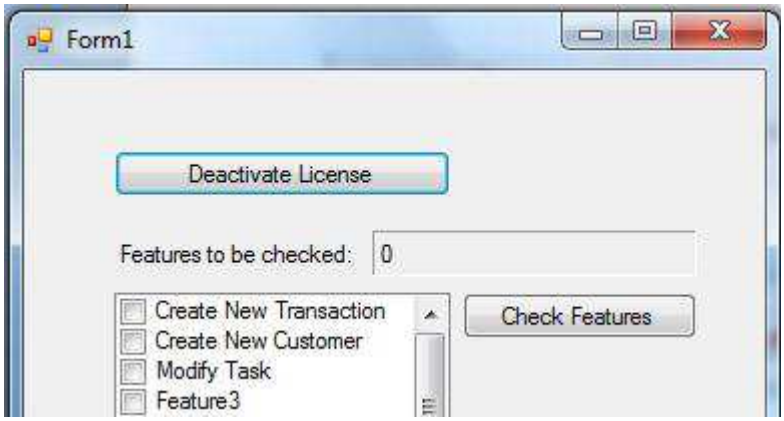


Click "Load license file":



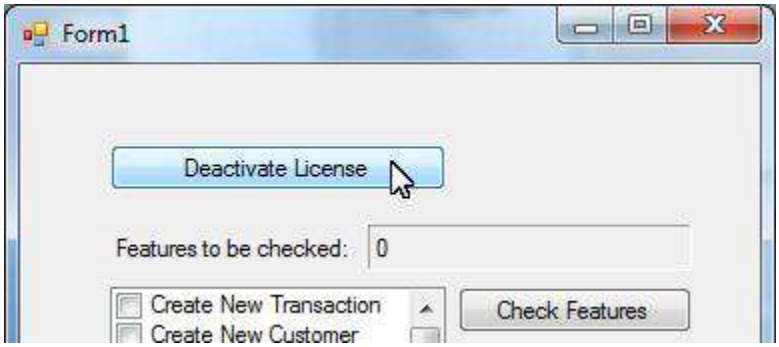


The software runs normally.

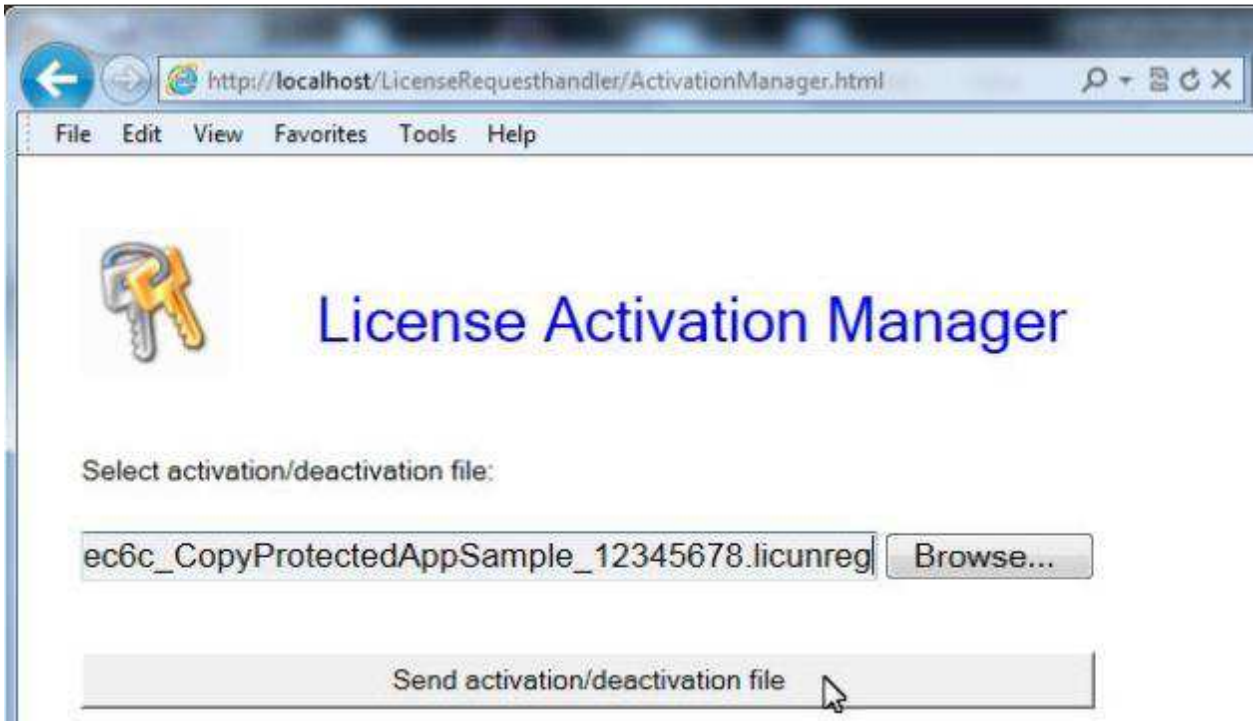


Send Deactivation File Manually

When deactivating a license, it also asks for sending a deactivation file to web server. If the computer does not have an internet connection then your user needs to copy the deactivation file to a computer having an internet connection.



Your user uses a web browser to open your web page ActivationManager.html and load the deactivation file to it; click “Send activation/deactivation file”:



Your web server processes the deactivation file. Your license management plug-in records the deactivation. A message is displayed to your user:

Modify Web Pages

You may want to make cosmetic changes to web pages. Especially, ActivationManager.html is used by your customers; you may want to make it look consistent with your other company web pages.

You may modify ActivationManager.html and Deactivate.html, as long as you do not modify the existing JavaScript code and do not remove HTML elements which are used by the existing JavaScript code.

Feedback

Please send your suggestions, bug reports, and requests to support@limnor.com. Thanks!