

Kentico allows developers to add various types of custom functionality organized within [modules](#). You may wish to provide these features to customers for a price. One way to achieve this is via licensing.

We provide tools and functionality necessary to implement licensing for custom Kentico modules – a license management application for the Kentico administration interface and an API for working with licenses in the code of custom modules.

To implement licensing for custom modules, you need to:

- [Construct licenses for your custom module](#)
- [Add license checks to the implementation of your custom module](#)
- [Add and manage licenses on all Kentico instances using the custom module](#)

To help you with the implementation of module licensing, the Kentico API provides the **ModuleLicensesHelper** class in the *CMS.ModuleLicenses* namespace.

The **ModuleLicensesHelper** class exposes the following methods for working with licenses:

- **void GenerateKeyPair(out string privateKey, out string publicKey)**  
Creates a new private and public key pair for generating and validating module licenses. Created keys are encoded in Base64.
- **string CreateModuleLicense(string licenseData, string privateKey)**  
Creates a module license containing license data and a signature based on the given private key.
- **IEnumerable<string> GetValidModuleLicenses(string moduleName, string publicKey)**  
Returns a collection of module license data with cryptographically valid signatures for a given module.

## Constructing licenses

### Generating cryptographic key pairs

Before you can construct your licenses, you need to generate a cryptographic key pair that will help secure your licenses against malicious tampering. You can generate a key pair via the **GenerateKeyPair()** method. For example:

```
using CMS.ModuleLicenses;

...

// A string variable for the private key component
string privateKey = "";

// A string variable for the public key component
string publicKey = "";

// Generates the private and public key components and saves them into privateKey and
publicKey respectively
ModuleLicensesHelper.GenerateKeyPair(out privateKey, out publicKey);
```

The **privateKey** and **publicKey** variables now contain the respective keys that can be used to [generate module licenses](#).



**Important:** If you are implementing module licensing for multiple custom modules, we recommend generating a different cryptographic key pair for each module.

## Generating licenses

To generate license keys for your module, use the **CreateModuleLicense()** method. The method requires the **private key** generated by the [GenerateKeyPair method](#) and **license data** as its arguments.

*License data* is a string containing arbitrary data used for license checks in the custom module. The license data may, for example, contain the username of the user for whom the license is intended and the license's expiration date.



Any data accessible from a Kentico instance can be used as part of the license data. For example, the **SiteContext.CurrentSite** property available in the *CMS.SiteProvider* namespace can be used with corresponding checks in the custom module to allow access to the module only to a user signed in to a particular site.

License generating example:

```
using CMS.ModuleLicenses;

...

// A string variable containing the generated private key
string privateKey = "generated private key";

// A string variable containing sample license data. Username and an expiration date
are used in this example.
string licenseData = "USER:administrator\r\nEXPIRATION:12/31/2030";

// Creates a module license containing the license data and a signature (hash of the
license data)
string license = ModuleLicensesHelper.CreateModuleLicense(licenseData, privateKey);
```

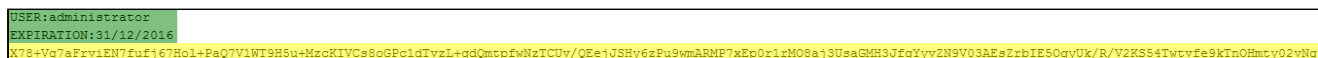
The **license** variable contains a valid module license that can be [added to a Kentico instance](#) via the **Module licenses** application.

## Generated licenses

Each generated license has the following format:

- **license data** – a string containing arbitrary data used for license checks in the custom module.
- **signature** – a hash of the *license data*. Generated and appended automatically to license data upon the license creation.

A sample license is shown in the image below. Highlighted green is the **license data**, and highlighted yellow is the **signature**.



```
USER:administrator
EXPIRATION:31/12/2016
X78+Yg7aFrviEN7fuEj67Hol+PaQ7V1WT9H5u+MzcRIVCs8cGPoldIvzL+qdQmcpfwNzTCUv/QEejJSHy6zPu9wmARMP7xEp0r1zMO8aj3U5aGMH3JfqYyvZN9V03AEs2zbIE5OqyUk/R/V2K55+Iwcvfe9kTnOHmty02vNg
```

## Implementing license checks in custom modules

After you have generated your licenses, you need to implement the appropriate license checks within your module's code. Kentico only provides an API for creating licenses and retrieving license data. Any license checking functionality on the code level must be implemented by the module's authors.

## Retrieving valid license data

To retrieve valid license data belonging to a custom module, call the **GetValidModuleLicenses()** method.



**Note:** The **GetValidModuleLicenses()** method requires the public key as one of its arguments. We recommend declaring the public key as a constant within the code of your custom module.

For example:

```
using CMS.ModuleLicenses;

...

// A string variable containing the generated public key
string publicKey = "generated public key";

// Code name of the custom module for which you want to retrieve the license data
string moduleCodeName = "LicensedModule";

// Returns license data for the given module as a list of strings
IEnumerable<string> licenses = ModuleLicensesHelper.GetValidModuleLicenses
(moduleCodeName, publicKey);
```

The **licenses** variable contains all valid license data belonging to a given module. You can use the retrieved license data to perform license checks in your module. See the [example below](#) for a general outline of the process.

### Example – Performing module license checks

The following example outlines a possible implementation of the license checking functionality on a simple scenario – a user trying to access a licensed module. For the purpose of this example, assume the license data is in format "USER:username":

1. [Retrieve module license data](#) via the **GetValidModuleLicenses()** method.
2. Create a method that iterates over the license data to find whether a license exists for the user trying to access the module.

```
private static bool IsUserLicensed(IEnumerable<string> licenses)
{
    // Iterates over all license data for the given module
    foreach( string license in licenses )
    {
        // Checks if the license is valid for the current user
        if (license.Contains("USER:" + MembershipContext.AuthenticatedUser.
UserName + "\r\n"))
        {
            return true;
        }
    }
    return false;
}
```

3. Call the method in places where you want to prevent access by unlicensed users (for example within the methods of the module's *InfoProvider* classes, or in code triggered by actions in the module's UI).

```
// Verifies whether the current user has a valid license in the "Module licenses"
application
if (!IsUserLicensed(licenses))
{
    // Custom logic covering cases where no valid license exists for the
current user
}
```

This example is not complete and is only meant as a general outline of license check implementations.

## Managing licenses for custom modules



Licenses need to be added via the **Module licenses** application in Kentico, which provides a single unified interface for displaying, inserting, and removing licenses for all custom modules in a Kentico instance.

All customers using custom modules with license checks need to use the *Module licenses* application to add valid licenses.

### Adding a license for a module

To add a generated license for a custom module:

1. Open the **Module licenses** application
2. Click **Add license**.
3. Select the module from the **License for module** drop-down list.
4. Copy the license data text into the **Module license** text area.
5. Click **Save**.

You can manage all added licenses in the **Module licenses** application. The **Edit** () and **Delete** () actions allow you to modify the corresponding license.