Dotcom Software Solutions Ltd

User Manual

Licence Master

Author:	Paul Taylor	File:	User Manual.doc
Date created:	20 July 2010	Source Safe	$\DnnDev\Modules\Licence
Last updated:	04 October 2012	Project:	Master

Confidentiality Statement

Revision History

Version No	Date	Author	Purpose
1	20/07/2010	paul	Created
2	26/07/2010	paul	Revised
3	24/08/2010	paul	Additional information on installation
4	02/11/2011	paul	New Client Library methods
5	27/11/2011	paul	Simplified client integration model and token renewals
6	01/03/2012	paul	v2.1 revisions
7	04/10/2012	paul	v2.2 revisions

Table of Contents

1. Introduction

1.1. What is Licence Master

Licence Master is a licence management application for software vendors and developers who wish to protect web applications from unauthorised use. It is an extension to the DotNetNuke (DNN) content management system, and as such, must be installed in a DNN site. It allows the vendor to manage licences for multiple applications and supports feature and version licensing as well as conventional licence agreements.

The business processes involved in managing licences for software products typically involve a complex set of interactions between the vendor and licensee, client application and a central licence management system. Licence Master co-ordinates and supervises these processes, presenting them to both the licence administrator and the licensee as a set of straightforward, easy-to-follow steps.

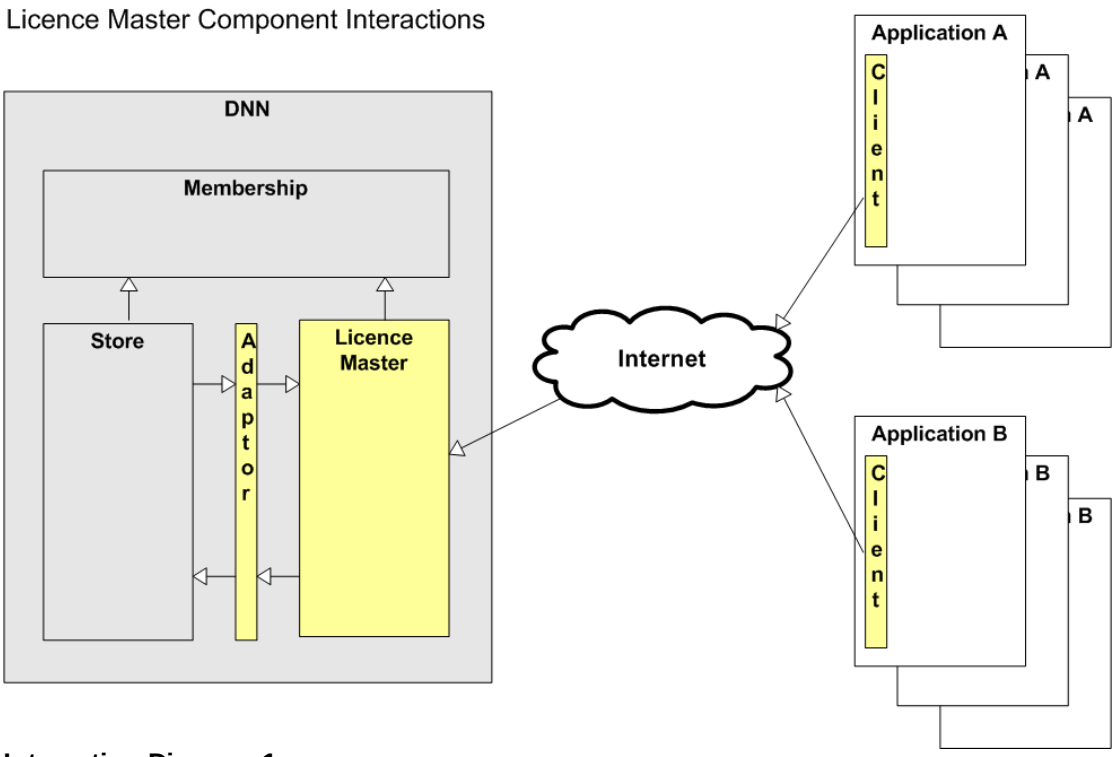
Licence Master enables automated licensing processes, and with it, more robust service, greater customer satisfaction and a significant saving in administrative effort. Yet it also offers flexibility; both in licensing policy and in the ability to intervene manually in the processes when required.

1.2. Who should read this manual?

This manual provides a detailed account of how Licence Master works. It also provides the technical information necessary to integrate an application with Licence Master.

If your company has chosen Licence Master as its licence management platform and you are responsible for either implementing it or for integrating applications with it, you should read this manual.

2. Operational Overview



Interaction Diagram 1

2.1. Licence Master Components

2.1.1. Administration Module

This module provides features for generating, renewing, upgrading and managing licences, for defining e-commerce policy, for reviewing renewal tokens (see below), and for defining the applications that you want to licence.

2.1.2. Registration Module

This module provides pages for registering, upgrading and renewing a licence. The licensee is required to create an account, or log into an existing one, to register the licence.

2.1.3. Response Processing Module

This module provides a landing page for encoded links from Licence Master notification emails. These are sent to the licensee to verify his email address or to approve the relocation of the site's registered domain. See Notification Emails below for further information.

2.1.4. Web Services

Licence Master Web Services provide the interface by which client applications communicate with it. However, client applications do not interact directly with the web services; instead they use the Client API component, described in *Client API*, below.

2.1.5. Client API

Client applications interact with Licence Master by means of a client API. The API is a .NET assembly that provides a range of functionality to the application including:

- Validating a licence
- Sending licensee notification emails
- Obtaining registration and download urls for the application

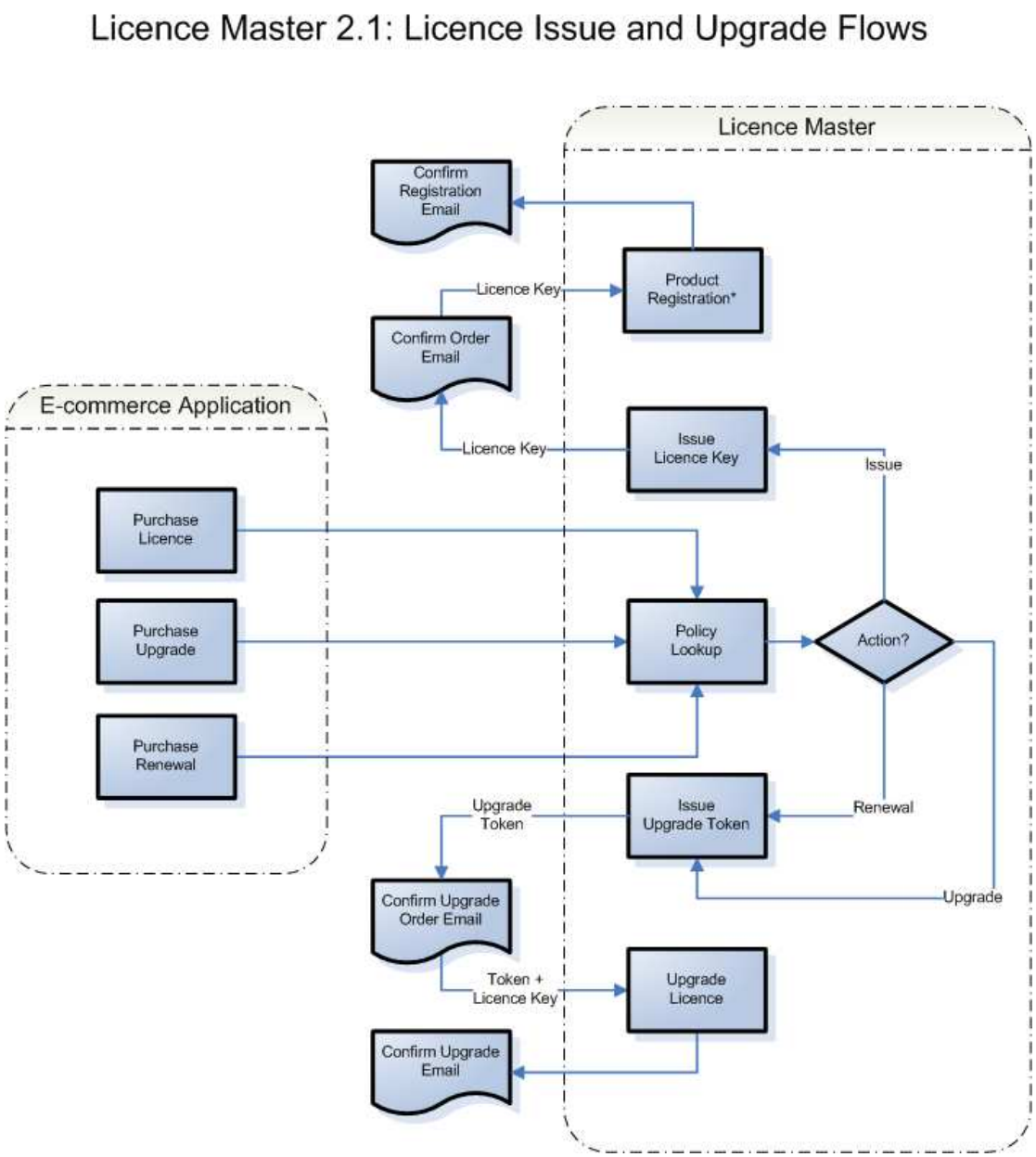
- Detecting which application features have been licensed (see Feature Licensing below)
- Obtaining the operating domain of the Licence Master application with which the client application is configured to communicate

For more information, see the section *Integrating Applications with Licence Master*, below.

2.1.6. e-commerce API

Licence Master's e-commerce API may be used by store modules installed in the same DNN instance to obtain licence keys during a product sale process. It enables automatic processing of licence issue, upgrades and renewals.

2.2. Process Diagram



2.3. Licence Issue and Registration Processes

The registration process involves the licensee completing four steps, following guidance from Licence Master and the client application[1]:

1. Obtain a licence
2. Install licence key in client application
3. Register licence
4. Verify email address

Two alternative issue processes are supported.

If Licence Master is integrated with a DotNetNuke e-commerce module, no intervention is required by a licence administrator in this process. The store module can order the licence key directly from Licence Master when completing the sales transaction.

If an integrated e-commerce module is not available, the order process occurs manually, for example by telephone or email. In this case, a licence administrator must generate the licences requested and send them to the customer by email.

2.4. Licence Upgrade and Renewal Processes

The licence upgrade process allows a licensee to enhance a licence to support a later version or superior edition of an application. It may also be used to extend a licence's validity period or to increase the number of domains that it is valid for.

The licence renewal process extends a licence's validity period and is only applicable to applications that use indefinite licensing (see *Licence Duration*, below).

Licence Master supports three different upgrade and renewal processes.

If Licence Master is integrated with an e-commerce module that allows the customer to enter a licence key during the purchase process, the licence may be upgraded or renewed directly, via the e-commerce API. No intervention is required in this process by a licence administrator.

Most e-commerce modules do not support the submission of custom data when buying products. For such cases, Licence Master supports an alternative renewal process based on tokens. In this process, the customer purchases a licence upgrade or renewal, but without specifying the licence key to be renewed. Licence Master issues a renewal token and sends it to the customer by email. The email contains a coded link to a page on the Licence Master server where the customer is prompted to enter the key of the licence to be renewed. No intervention is required in this process by a licence administrator.

Finally, if no integrated e-commerce module is available, the renewal process occurs manually, by telephone or email. A licence administrator enters the key of the licence to be renewed, and the duration of the renewal, and a confirmation email is sent to the licensee.

See *Registration Process Diagram*, below for an illustration of the registration process.

2.5. Email Notifications

Licence Master communicates with licensees by means of automated email notifications. There are twelve emails in total, detailed in the table below.

The Client API provides a method to instruct Licence Master to send any notifications that correspond to the current status of an application installation. These are highlighted in the table below.

[1] It is good practice for client applications to provide the licensee with a link to the product registration page when they install the licence key.

Notification	When sent	Sent by Client API for Status
Order Confirmation	When ordering a licence using the e-commerce API	n/a
Registration Confirmation	After registering a licence and verifying account email address	n/a
Relocation Confirmation	When an attempt is made to re-register a licence key at a domain for which the licence is not already registered and the licence covers only a single domain	n/a
Renewal Confirmation	After renewing a licence using the e-commerce API	n/a
Upgrade Confirmation	After upgrading a licence using the e-commerce API	n/a
Licence Expiry	When the client application requests it	Licence Expired
Relocation Warning	When the client application requests it	Location Invalid
Renewal Reminder	Sent by the Licence Expiry Notification Service	n/a
Send Renewal Token	When ordering a licence renewal using the e-commerce API	n/a
Send Upgrade Token	When ordering a licence upgrade using the e-commerce API	n/a
Email Verification Reminder	When the licensee re-registers a licence key and has not yet verified his email address; or when the client application requests it	Email Verification Pending
Email Verification Request	When a licensee registers a licence key	n/a

3. Download and Licensing

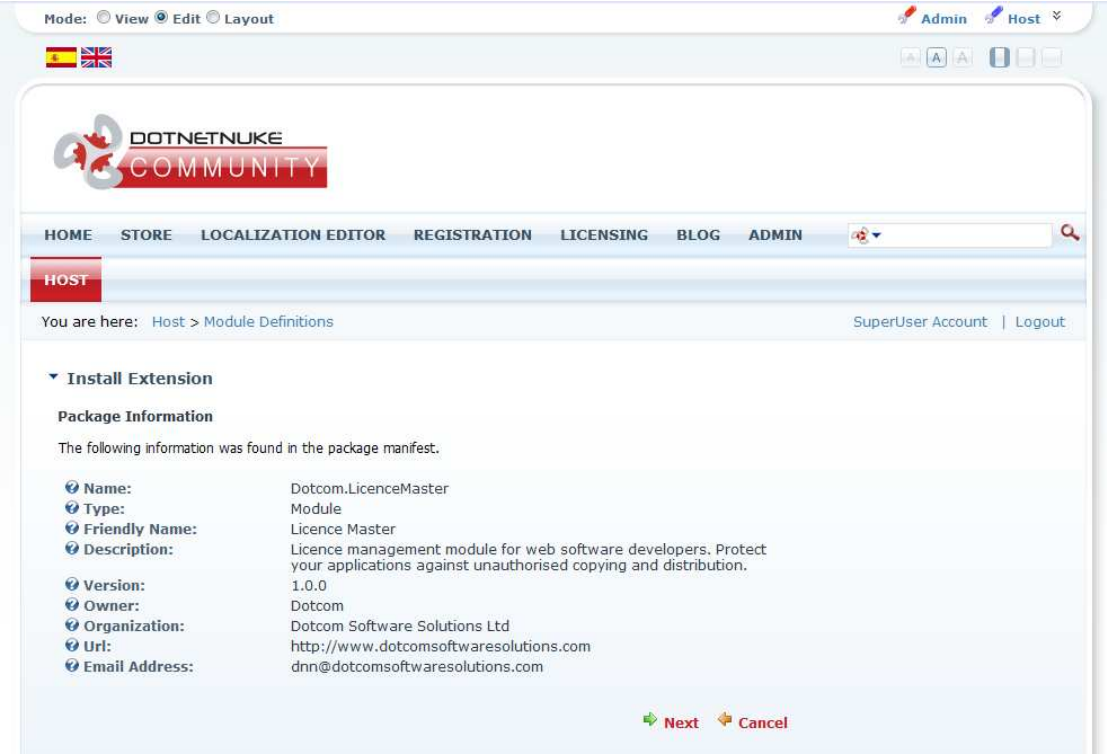
A free trial version of Licence Master can be downloaded from the Dotcom website at: http://www.dotcomsoftwaresolutions.com/en/Products/Downloads.aspx.

For licensing terms, please refer to: http://www.dotcomsoftwaresolutions.com/en/Resources/Licensing.aspx.

4. Installation

Licence Master is installed like any other DotNetNuke module.

1. Log on as a super user
2. Navigate to Host -> Module Definitions
3. Put the page in Edit mode
4. At the bottom of the page, click the Install Module button
5. Browse to find the installation zip file on your local machine
6. Follow the module installation wizard



Once you have successfully installed the package, add Licence Master to a page in your site. Follow the steps below:

1. Add a new page to your site. This will contain Licence Master's administration control panel, so you should restrict access to members of the Administrator role only.
2. Navigate to the new page
3. Ensure that the page is in Edit mode
4. From the modules list, select Licence Master
5. Select the panel where you want the module displayed
6. Click Add module

Before you can do anything with Licence Master, you need to define at least one client application. When you add the module to the page, Licence Master will prompt you to do so.

1. Click on the Add Application button
2. Enter the application details, and click Update
3. If you defined a Feature Licensed application, click Cancel
4. Click Return to go back to the main module display

For further information on configuring Applications see *Defining Applications*, below.

Licence Master contains three components, and you will now see these displayed on the page. The first module is for product registration and renewal. The second module is for

licensee email validation and for relocating a licence's registered domain. You will normally want to display these on separate pages, so that you can control access to them in an appropriate way. To achieve this, follow the steps below:

1. Create a new page for the Registration module. Grant view access to all users
2. Create a new page for the Validation module. Grant view access to all registered users
3. Navigate to the Licence Master Administration page
4. Open the settings page for the Licence Master Registration and Validation modules in turn. Use the Advanced Page Settings to move each module to its respective page

5. Configuration

5.1. Defining Applications

Licence Master enables the vendor to manage licences for multiple applications. The administration module provides tools for defining those applications and the licence policy for each of them.

5.1.1. Application Identifiers

Two identifiers are used to represent an application: an internal numeric one; and a Globally Unique Identifier (GUID) which the client application uses to identify itself in interactions with Licence Master.

5.1.2. Application Product Identifiers

An application has two product identifiers. These are used to relate it to licence products that are available in an associated e-commerce module (see 7*Integrating DNN Store Modules with Licence Master*, below). One identifier is used to identify a licence sale, the other a licence renewal.

5.1.3. Licence Types

Two types of licence are supported:

- Standard licences support a conventional licence agreement in which the customer is granted use of the whole product for the period of the licence
- Featured licences allow the vendor to enable or disable specific features of the application, without the need to build and supply separate deployment packages. This type of licence is used to support applications that are sold in multiple editions.

5.1.4. Licence Duration

Licence Master supports two types of licence duration:

- Indefinite, which grant unrestricted use of the application
- Subscription, which require renewal after the licence expiry date

Subscription licences may be issued with duration of one, two or three years.

5.1.5. Domains

Licences issued by Licence Master grant permission to operate an application on one or more domains. The licensee can register and use the application on as many domains as the licence covers.

5.1.6. Application Features

A list of features can be defined for applications with a Featured licence type. These features can be dynamically enabled and disabled by the application in accordance with information that it receives from Licence Master about the features licensed for a particular installation.

5.1.7. Application Editions

A list of application editions may also be defined for Feature licensed applications. Application editions represent a set of licensed features. The edition that the licensee selects when ordering a feature licence determines which features are licensed.

5.2. Defining E-commerce Policy

5.2.1. Overview

The purpose of the e-commerce policy feature is to allow Licence Master to translate incoming orders from an e-commerce module into specific licensing actions. These actions create new licences or transform existing ones according to a specification defined in the policy rule.

5.2.2. E-commerce Product Id

The Product Code is defined at application level. It is a code passed by an e-commerce module that is used to identify a Licence Master application. When integrating with the store.dotnetnuke.com e-commerce site for example, this will be the product's PackageId.

5.2.3. Product Option

Product Option is defined at policy rule level. It is a code passed by an e-commerce module that that is used to identify a policy rule. When integrating with the store.dotnetnuke.com e-commerce site for example, this will be the product option Id.

5.2.4. Licence Actions

Licence Master can execute three actions:

- Issue
- Renew
- Upgrade

5.2.5. Action Specification

A policy rule defines two sets of fields. The first defines criteria for the licences to which the rule may be applied. The second defines the values to which the licence's various characteristics will be upgraded. Some fields are available only for particular Actions, and some only for Featured applications, as summarised in the table below.

	Actions	**Featured Only**	**Description**
Valid For Edition	Upgrade / Renewal	Y	Edition of licences to which this rule may be applied
Valid For Version	Upgrade / Renewal	N	Version of licences to which this rule may be applied
Valid For Domains	Upgrade / Renewal	N	Domains of licences to which this rule may be applied
Edition	Issue / Upgrade	Y	The application edition for which the licence will be created or to which it will be upgraded
Version	Issue / Upgrade	N	The application version for which the licence will be created or to which it will be upgraded
Domains	Issue / Upgrade	N	The number of domains for which the licence will be created or by which it will be incremented
Duration	Issue / Upgrade / Renewal	N	The validity period in months for which the licence will be created or by which it will be extended

5.3. DNN Security Configuration

Licence Master takes advantage of DotNetNuke's security configuration system to restrict access to its various pages. Although the modules can be secured in any way that suits the vendor's purpose, the normal security configuration is detailed in the sections that follow.

5.3.1. Administration Module

As the administration module enables the user to browse, issue and renew licences and to define applications, access to it should only be granted to site administrators or equivalent roles.

5.3.2. Registration Module

Some licensees may need to access the page provided by the registration module before having a user account at the site. For that reason, view access to the registration module should be granted to all users.

5.3.3. Validation Module

The validation module allows the licensee to verify his email address and to authorise Licence Master to re-register a licence for a different operating domain. For that reason, access to it should be controlled by authentication. To achieve this, view access should be granted to the DNN Registered Users role.

6. Integrating Applications with Licence Master

Integrating your applications with Licence Master is a simple ten-step task that can be completed in less than 5 minutes.

Once Licence Master functionality is available to your application, your application can use it to place restrictions on its own use in any way that you wish.

There are many possibilities, and some of the more common ones are described in the section *Typical Client Application Processing*, below.

6.1. Application Requirements

Client applications need to make use of .NET assemblies to communicate with Licence Master. The licensing model also presupposes an application that operates on an internet domain. Taken together, these two requirements mean that Licence Master is suitable for use with ASP.NET applications.

Licence Master is ideal for managing licensing for DotNetNuke modules but can also be used with any ASP.NET application.

6.2. Preparing the Application

To enable your application to communicate with Licence Master, you need to follow the four steps below. In your application development environment:

1. Copy Dotcom.Modules.LicenceMaster.Client.dll (included in the Licence Master installer zip file) to the bin folder of your application
2. Add a reference to it to your development project or website
3. Add a Guid attribute to your application assembly (in Assembly.cs or Assembly.vb). The value of the Guid must be the same as that defined for the application in the Administration module
4. Add two ServiceUrl attributes to your assembly. These must provide valid urls for the web services at the Licence Master installation that will service the application (see below):

```
using Dotcom.Modules.LicenceMaster.Client;

[assembly: Guid("cc9e1354-75a9-439e-8c7f-7013e12a85d1")]
[assembly: ServiceUrl(ServiceType.Installation,
"http://www.mycompany.com/DesktopModules/Dotcom.LicenceMaster/Installation.asmx
")]
[assembly: ServiceUrl(ServiceType.Application,
"http://www.mycompany.com/DesktopModules/Dotcom.LicenceMaster/Application.asmx"
)]
```

Your application is now ready to communicate with Licence Master at your chosen domain.

6.3. Basic Integration Techniques for DotNetNuke modules

Licence Master comes with a class library that makes integrating a DotNetNuke module a quick and effortless task.

1. Copy Dotcom.Modules.LicenceMaster.Client.Dnn.dll (included in the Licence Master installer zip file) to the bin folder of your DNN website
2. Set a reference to it in your module development project

3. Change all your module controls so that they inherit from the class `LicenceMasterClientModuleBase`. Change all settings controls so that they inherit from the class `LicenceMasterClientModuleSettingsBase`.[2]
4. Implement the LicenceKey property for each module or settings control. This property must return the installed licence key, typically retrieved from module settings (see example below).
5. Add a LicenceKeyControl server control to your settings page
6. Write an event hander for the control's LicenceStatusRequired event that retrieves the status of the licence and returns it to the control (see example below).

```
protected override string LicenceKey
{
	get
	{
		return (string)Settings["LicenceKey"];
	}
}

protected void lkcLicenceKey_LicenceStatusRequired(object sender,
                                    LicenceStatusRequiredEventArgs args)
{
	LicensingUtility lu = new LicensingUtility(
                            Assembly.GetExecutingAssembly(),
                            args.LicenceKey, Request.Url.Host);
	args.Status = lu.GetInstallationStatus();
}
```

Your module is now integrated with by Licence Master. By default, it will display a licensing reminder banner at the top of the page until a valid licence key has been installed and registered. You may also optionally customise the behaviour of these components. Download the sample Licence Master integrated DotNetNuke module from our website for example customisations:

http://www.dotcomsoftwaresolutions.com/en/Support/Downloads.aspx

6.4. Advanced Integration with the Licence Master Client API

The section above describes how your application can quickly and easily implement basic licence validation functionality. However, the Licence Master Client API provides access to detailed information about an installed licence that your application can use to implement custom restrictions according to the licence's status.

Note that the techniques in this section can be used both for DotNetNuke modules and for ASP.NET applications.

The Client API provides a single class with the methods that your application needs to interact with Licence Master. To use it, declare an instance of the `LicensingUtility` class, passing it the application assembly, the installation licence key, and the site host:

```
using System.Reflection;
using Dotcom.Modules.LicenceMaster.Client;

LicensingUtility lu = new LicensingUtility(Assembly.GetExecutingAssembly(),
licenceKey, Request.Url.Host);
```

[2] If your module features several controls that share common functionality, it may be more convenient to create your own module or settings control base classes that in turn inherit from `LicenceMasterClientModuleBase` or `LicenceMasterClientModuleSettingsBase` as applicable.

Note that validating the status of the site depends on the application having access to the licence key. Until the licensee has installed the licence key, you cannot pass it to LicenceMaster to determine whether the installation is valid and therefore you cannot create an instance of the `LicensingUtility` class. Your application must implement any necessary restrictions on how the site may be used before the licence key is installed.

The application is now ready to retrieve licensing information from LicenceMaster. The methods that you use are listed in the table below.

Method	Arguments	Description
CanConnect	None	Returns true if the client application is able to communicate with Licence Master
GetConnectionStatus	None	Returns an HttpStatusCode value indicating the status code return when attempting to communicate with Licence Master
ConnectionStatusValid	None	Returns true if the connection status can be treated as valid. This depends on the status codes that the client application configures to be tolerated (see section 6.8 below).
WillTolerateConnectionStatus	HttpStatusCode status	Returns true if the given status code is amongst those that the client application configures to be tolerated (see section 6.8 below).
InstallationIsValid	None	Returns true if the licence key has been registered and the licensee has verified his email address
GetInstallationStatus	None	Returns the current status of the product installation
GetExpiryDate	None	Returns the expiry date of the licence
GetApplicationRegistrationUrl	None	Returns the url of the page configured as the product registration page
GetApplicationDownloadUrl	None	Returns the url of the page from which the product can be downloaded
SendNotification	None	Sends an email notification to the licensee alerting him to any issues with the product licence.
FeatureLicenced	string licenceKey byte featureId	Used with Feature licences. Returns true if the specified feature is enabled for the specified licence key
AnyFeatureLicensed	string licenceKey byte[] featureIds	Used with Feature licences. Returns true if any of the specified features are enabled for the specified licence key
AllFeaturesLicensed	string licenceKey byte[] featureIds	Used with Feature licences. Returns true if all of the specified features are enabled for the specified licence key

6.5. Typical Client Application Processing

With Licence Master functionality available, your application can choose to place restrictions on its own operation, depending on licence status. The aim is to encourage the user to licence the product. There are many different ways of doing this. Some of the common ones are reviewed below.

Note that the DNN control base class LicenceMasterClientModuleBase *defines properties and methods to assist with these tasks.*

6.5.1. *Display a trial banner*

Displaying a trial banner is perhaps the friendliest approach, though intrusive enough to deter use of the unlicensed use of the application. The banner is displayed when the product is not licensed, or when the licence status is not satisfactory.

The banner should explain its own appearance, and provide information about how to remove it (for example by giving a link to a website where licences can be obtained).

The banner markup and content should be compiled into the application (e.g. as a string constant or embedded resource) and should not contain references to external stylesheets. This will prevent it from being disabled by the user. If you need to localize the text, you can do so in normal resource files, but it is advisable to maintain default banner markup in your "fallback" language compiled into the application in case a user should delete the relevant entries from the resource files.

If you want to display the banner on all application pages when the site is not correctly licensed, you have two choices.

You can either inject it during the HttpApplication.BeginRequest event (exposed in global.asax):

```
protected void Application_BeginRequest(object sender, EventArgs e)
{
    LicensingUtility lu = new LicensingUtility(
                Assembly.GetExecutingAssembly(),
                licenceKey, Request.Url.Host);
    if (!lu.InstallationIsValid())
    {
        Response.Write(UnlicencedBannerText);
    }

}
```

Or, you can implement a base class for all your site pages, and inject the banner during the Page.Init event

```
protected override void OnInit(EventArgs e)
{
    LicensingUtility lu = new LicensingUtility(
                Assembly.GetExecutingAssembly(),
                licenceKey, Request.Url.Host);
    if (!lu.InstallationIsValid())
    {
        Response.Write(UnlicencedBannerText);
    }
}
```

In DotNetNuke modules, you can use a similar approach: create a base class for your module controls and set the LicenceMasterClientModuleBase.BannerText property in the class constructor or during its Init event.

Important Note. Calling Licence Master on every page load is not necessary and may affect application performance. See the section *Caching Licence Status*, below.

6.5.2. *Send a notification to the licensee*

When your application has a licence key installed, and it has been registered, the licensee is known and so you can send notifications about licensing problems to him. You may decide to send such notifications to the licensee whenever the application starts. It is also worth considering whether to limit the frequency with which notifications are sent by another means to avoid sending too many.

To send a notification about any issues with site licence, simply call the LicensingUtility.SendNotification() method. Notifications are only sent when there is an issue to report.

```
LicensingUtility lu = new LicensingUtility(
            Assembly.GetExecutingAssembly(),
            licenceKey, Request.Url.Host);
lu.SendNotification();
```

6.5.3. Disable application features

Another way of encouraging the user to license the product is to restrict the use of key application features when the licence status is not satisfactory.

```
LicensingUtility lu = new LicensingUtility(
            Assembly.GetExecutingAssembly(),
            licenceKey, Request.Url.Host);
if (!lu.InstallationIsValid())
{
    DisableShoppingCart();
}
```

6.5.4. Disable application

Disabling an application is the most drastic option, though it may be appropriate when other options are not available. It is recommended to display a page to the user (likely to be the licensee in the first instance) with information about how to resolve the problem. That may include contacting your licensing department or administrator.

A typical implementation of this approach involves detecting the status of the installation during application initialisation (e.g. in the HttpApplication.Init event, exposed by global.asax), and if it is not satisfactory, redirecting the user to an appropriate url.

6.6. Caching Licence Status

Once a site has a licence installed and registered, the status is not going to change very often. In fact, it will only change if the licensee reinstalls the application on a different domain or if the licence expires.

It is therefore not necessary to call Licence Master every time a page loads (for example) as the status is very unlikely to have changed. Unnecessary web service calls may impact application performance. To avoid this, you may choose to cache the licence status. The cached value can be configured to expire after a period that you decide, so that for example, the application will check with Licence Master at least once a day to see if the licence has expired.

However, while the installation is not licensed and registered, the status may change frequently, such as when the licensee installs a licence key, when he registers the product, and when he verifies his email address. As a result, you might consider *not* caching the licence status *until* the process is fully completed.

Also, remember to remove the cached licence status if the user does anything with the application that would affect it, such as removing or changing the licence key.

6.6.1. Examples

Both DotNetNuke and ASP.NET provide caching facilities. For DotNetNuke, the LicenceMasterClientModuleBase class has a cached ProductInstallationValid method that provides a simple way of determining if the module installation is valid. The class also

exposes a LicenceCache object that provides more cached licensing methods that your modules can use if necessary.

The example below shows how to the ASP.NET Cache class.

ASP.NET

```
using System.Web.Caching;

        public bool InstallationIsValid()
        {
            Cache cache = new Cache();
            string cacheKey = string.Format("MyApp:InstallationValid:{0}",
                                            licenceKey);
            bool valid = false;
            object cachedValue = cache[cacheKey];
            if (cachedValue == null)
            {
                valid = LicensingUtility.InstallationIsValid();
                if (valid)
                {
                    //Cache validity only if true
                    cache.Add(cacheKey, valid, null, DateTime.Now.AddDays(1),
                              Cache.NoSlidingExpiration,
                              CacheItemPriority.Normal, null);
                }
            }
            else
            {
                valid = (bool)cachedValue;
            }
            return valid;
        }
```

6.7. Feature Licences

Licence Master supports Feature Licensing. That means that as well as being able to find out the status of an installation, your applications can find out which application features have been enabled for a particular licence.

6.7.1. Defining Application Features

The Administration module provides a facility for defining a list of features for an application (see *Application Features*, above). You provide each feature with a numerical identifier. Your application uses the same identifiers to determine whether application features have been enabled for a licence.

For example, suppose that your application defines the following enumeration with values that correspond to the feature identifiers defined in Licence Master:

```
    internal enum Features: byte
    {
        Accounts = 0,
        Shop = 1,
        Basket = 2,
        Checkout = 3
    }
```

The application can determine whether the Accounts feature is enabled using this code:

```
    if (!LicensingUtility.FeatureLicensed(licenceKey, Features.Accounts))
```

```
{
    DisableCustomerAccounts();
}
```

6.8. Configurable Tolerance

The Client API incorporates a number of methods that can be used to determine the status of your application's connection to the Licence Master. Furthermore, your application can specify a list of status codes that it will tolerate in the event that it is unable to communicate with Licence Master. This gives you the means to ensure that your application continues to function if for some reason Licence Master is temporarily unavailable, due to say, a maintenance outage or an issue with your web hosting.

This is achieved by adding an extra attribute to your application assembly (in Assembly.cs or Assembly.vb):

```
[assembly: ToleratedStatusCode(new HttpStatusCode[]{ HttpStatusCode.OK,
HttpStatusCode.InternalServerError,
HttpStatusCode.NotFound,
HttpStatusCode.MethodNotAllowed })]
```

When the code returned by the GetConnectionStatus() method is included in list specified by the attribute, the ConnectionValid() method will return true, and WillTolerateStatusCode(status) will also return true. Note that the CanConnect() method will always return false unless the connection status is `HttpStatusCode.OK`.

If CanConnect() returns false, but ConnectionValid() return true, the InstallationIsValid() method will also return true and the GetInstallationStatus() method will return `InstallationStatus.Registered`. This means that your application will behave as if it is correctly licensed and registered.

If your application does not specify any status codes in the `ToleratedStatusCode` attribute, the InstallationIsValid() and GetInstallationStatus() methods will also return true and `InstallationStatus.Registered` respectively. This is for compatibility with previous releases of the Client API.

7. Integrating DNN Store Modules with Licence Master

7.1. Why Integrate?

Licence Master's e-commerce API makes a fully automated product sale process a possibility. When a customer purchases your application from an online store, as well as delivering the product to him, the store application can instruct Licence Master to issue and send a licence key to the buyer. Similarly, it can initiate the process to renew or upgrade an existing licence.

A site that integrates Licence Master and an e-commerce module reduces or eliminates the need for administrative intervention and provides a trouble-free sales process, enhancing customer satisfaction.

The customer also benefits in other ways. Licences purchased as part of a product sale may be registered on issue, removing part of the licensing and registration process for the customer.

7.2. Application Requirements

Licence Master is designed to work with DotNetNuke store modules. The close collaboration between store and licence management applications requires both to have access to common data such as customer information.

To integrate a store module with Licence Master, you need to have access to its source code. Many DotNetNuke store modules are freely available with source code; however, if the source code is not available for your chosen store, check with Dotcom to find if we have already developed an adaptor for the store in question (see the following section). If not, you might also contact the module distributor to see if they have developed an adaptor.

7.3. Dotcom Store Adaptors

Rather than integrate a store module with Licence Master yourself, you can install one of Dotcom's store adapters. Dotcom is the vendor of the store.dotnetnuke.com adaptor. It is also working with store module developers to produce adaptors for a number of popular DotNetNuke stores. We also undertake custom store integrations to order.

7.4. Preparing the Application

To enable a store module to interact with Licence Master, follow the steps below. In your application development environment:

1. Copy Dotcom.Modules.LicenceMaster.dll (included in the Licence Master installer zip file) to the bin folder of the e-commerce module
2. Add a reference to it in the store development project

7.5. Using the e-commerce API

The e-commerce API enables DNN store modules to order application licences and renewals from Licence Master as part of a product sale process. How this is implemented will vary from module to module, but the basic pattern is as follows.

First, each of the applications defined in Licence Master must be added as products to the store module. Secondly, you must define the e-commerce policy for each application (see *Defining E-commerce Policy*, above). For each policy rule, you define a product option in the store module.

When the store module receives confirmation of payment for an order, it usually performs some processing of the order, if only to mark its status as paid. Note that this may not always

occur at the point that the customer confirms the order. Sometimes it may occur later, in response to a payment confirmation (or denial) callback from a payment service provider.

As part of the confirmation order processing the store can be modified to check to see if any licence products require to be processed. When it finds a licence product, it can call Licence Master using data stored in the order at the time it was placed by the customer.

When ordering a licence or an upgrade using the API, you pass Licence Master the product identifier used by the store for the relevant application. You also pass the identifier of product option which represents the policy rule associated with the option.

Licence Master uses this information to identify the application and the policy rule to execute. For this to work, the same identifiers must be also be defined for the corresponding application and policy rule in Licence Master. The relevant fields are E-commerce Product ID and Product Option.

See the example processing in the code below:

```
    LicenceMasterController lmc = new LicenceMasterController();
    ApplicationInfo app = lmc.GetApplicationForLicenceProduct(packageId);
    if (app != null)
    {
        //Process order
        Licence licence = Licence.Create(app.ApplicationId);
        ArrayList licenceList;
        if (!optionID.HasValue)
        {
            //This is a simple issue with default parameters
            if (licence is FeatureLicence)
            {
                //a Feature Licenced application must have one or more editions
                //defined
                throw new
ApplicationException(string.Format(Localization.GetString("OptionNotFound.Error
Message", LocalResourceFile), optionID));
            }
            else
            {
                short? duration = (app.HasIndefiniteLicence) ? (short?)null :
12;
                licenceList = licence.Order(app.ApplicationId, quantity,
DateTime.Today, 1, duration, billToEmail);
            }
            RegisterLicences(licenceList, orderId, ac);
        }
        else
        {
            licenceList = licence.Order(packageId, optionID.Value, quantity,
billToEmail);
            if (licenceList != null)
            {
                //The licence list is populated for Issue actions
                RegisterLicences(licenceList, orderId, ac);
            }
        }
    }
    else
    {
        throw new
ApplicationException(string.Format(Localization.GetString("PackageNotFound.Erro
rMessage", LocalResourceFile), packageId));
    }
```

8. Support

Dotcom has produced a number of resources to provide support for Licence Master users.

8.1. Technical Documentation

Download user manuals from
http://www.dotcomsoftwaresolutions.com/en/Resources/Downloads.aspx

8.2. Forum

Please raise all support requests in the first instance on our support forums:
http://www.dotcomsoftwaresolutions.com/en/Resources/Forums.aspx

8.3. KnowledgeBase

The Licence Master is growing at:
http://www.dotcomsoftwaresolutions.com/en/Resources/KnowledgeBase.aspx

8.4. FAQs

Read the product FAQs at:
http://www.dotcomsoftwaresolutions.com/en/Resources/FAQs/Licence-Master-FAQ.aspx

9. Appendices

9.1. Registration Process Diagram