What's new in QLM 6.0

- New Metro style dashboard that presents a quick overview of the data in the system.
- New License Registration Wizard. The new wizard can be easily integrated in a .NET application or launched as a standalone executable for applications that are not using .NET (Protect your application tab).
- New card view type under Manage Keys. The data can now be presented as a Card view (see View Type drop down in the status bar).
- Improved QLM Web Service setup.
- New APi functions:
    - Get Volume Serial Number
    - Get CPU ID
    - Get MAC Address
    - Detect if running on HyperV, VMWare or VirtualBox
- Enhancements to QLM Application
    - Ability to check/uncheck all features when creating a license key.
    - Ability to set comments when creating a license key.
    - Ability to save default values when creating a license key.
    - The Sites/Test button validates that the encryption keys are correct.
- New QLM Cloud Edition that allows a company to manage products from multiple vendors.
    - All functionality related to Affiliates is now part of the Cloud Edition.
    - Ability to manage user accounts directly from QLM Console.
    - Improved Affiliates Portal.
    - See Cloud section in ribbon bar under Manage Keys.

Is this all? This is just a start. We will be gradually adding many more features to QLM v6. For more details about our development process, read this article.

Upgrade to QLM 6.0

The procedure below outlines the steps requires for users of earlier versions of QLM to upgrade to QLM 6.0.

QLM Express

To upgrade your application to use QLM 6.0, simply update that QLM DLLs that you ship with your application to the ones included with QLM 6.0.

Note that the name of the QLM redistributables has not changed. The QLM DLLs are: IsLicense50.dll, QlmLicenseLib.dll and QlmControls.dll.

QLM Professional

In addition to the steps outlined for QLM Express above, you should perform the following steps to upgrade to QLM 6.0.

1. Update your web service with the latest DLLs from the DeployToServer folder, or alternatively, run the qlmwebsvcsetup.exe on your web server.

2. If you decide to perform a manual upgrade of the web service, you need to update the web.config file of the QLM web service. The QLM 6.0 web.config file has changed significantly and as such, you should not attempt to patch your existing web.config file but rather, replace your web.config file with the QLM 6.0 version and then update the following settings:

- CommunicationEncryptionKey
- AdminEncryptionKey
- ConnectionStrings

Note that you can optionally keep the QLM 5 web service as is and install the QLM 6 web service in a different virtual directory.

2. After you have upgraded your web service, perform a database upgrade from the QLM console as follows:

- Start the QLM Console
- Click the Manager Keys tab
- Click on the Sites button
- Select your site and click on the Upgrade Dabase Schema button. Note that this step be performed after you have updated the DLLs of the web service to the latest version.

The QLM 6 Web Service Application API is fully compatible with previous clients. If you have existing customers using your application with QLM 5 DLLs, those clients can continue to communicate with a QLM 6 web service.

The QLM 6 Console application however is not backwards compatible and requires a QLM 6 web service.

Overview

Quick License Manager (QLM) is the ideal tool for software vendors who need to add licensing support to their products.

QLM uses several measures to defend the integrity of your software product. These include:

Tracking the date of first installation: The date on which your software was first run on the user's machine is permanently recorded by QLM. Even if the user were to uninstall the product then re-install it at a later date, the original trial period would remain in effect.

Monitoring the system date for tampering: If the user sets the system date back manually to defeat the evaluation period, QLM detects the tampering and disables the license.

Creating computer-bound keys: A license key created under QLM can be bound to a single computer. Attempts to use the same key on other computers will not succeed.

Safeguarding the licensing DLL: An easy-to-use mechanism allows you to verify at run-time that the code in QLM's licensing library has not been surreptitiously modified.

Using asymmetric encryption: License keys created under QLM are encoded with an asymmetric encryption algorithm based on public and private keys.

Overview

QuickLicenseManager is the ideal tool for software vendors who need to quickly add license key support to their products. Both evaluation and permanent license keys can be generated.

Adding licensing support to your application is a 3 step process:

- Define your products.
- Generate license keys.
- Modify your application to capture and validate a license key.

The simplest way to get started with QLM is to launch the **Getting Started Wizard** by clicking on the **Get Started** tab then selecting **Guide Me**.

For a quick overview of QLM, take a look at our online demos below.

Define products

The first step in creating license keys for your products is to define each product. Click on the **Define Products** tab and then click on **New.** Enter the Product Name, the Major and Minor version of the product. The Product ID and the GUID fields are automatically generated. You will need these values when you use the API to validate license keys.

For more details about defining products, refer to the Define Products section in the User Interface Guide.

Generate license keys

Once you have defined your products, you can generate license keys for each product. There are 2 types of license keys: permanent, and evaluation keys. Evaluation keys can specify the duration of the evaluation, an expiry date or both. In case both an expiry date and a duration period are specified, the key will expire when any of the 2 criteria are met.

To generate license keys using **QLM Express**, click on the **Generate Keys** tab. Select a Product, the number of keys to generate, the type of keys to generate and then click on the **Generate** button. **Quick License Manager** typically generates unique license keys, however, under some circumstances it may be possible that a key value is duplicated. **Quick License Manager Express** does not store generated keys. It is up to the user to track the generated keys if needed. For more details about creating license keys with QLM Express, refer to the Create Licenses section in the User Interface Guide.

To generate license keys using **QLM Pro** or **Enterprise**, click on the **Manage Keys** tab, then click on the *Create* button. Select a Product, the number of keys to generate, the type of keys to generate and then click on the *OK* button. For more details about creating license keys, refer to the Create Licenses section in the User Interface Guide.

Modify your application

To capture and validate a license key in your code, use the Protect your application to generate the source code required to add licensing to your application.

Online Demos


QLM Pro Demo


QLM Express Demo


Online Activation Demo


FastSpring (eCommerce provider) Integration Demo

For more online demos, refer to our web site or run the Get Started Wizard in the QLM Console.

License key types

Quick License Manager supports 5 different types of license keys:

- **Evaluation or Subscription License Keys:** Evaluation keys can specify the duration of the evaluation, an expiry date or both. In case both an expiry date and a duration are specified, the license will be flagged as expired when the duration period is exceeded or the expiration date has passed.
- **Not Computer Bound or Generic:** Generic keys are not bound to a specific computer. The advantage of this type of key is that you can pre-generate a large set of keys and just distribute them to any customer. The disadvantage is that any user who gets access to a key is able to use that key on any computer.
- **Bound to Computer Name:** Computer Name bound keys are keys that are bound to a specific computer by encrypting the name of the computer in the license key. Computer Name bound keys can be generated from the Quick License Manager user interface or programmatically using the Quick License Manager API (CreateLicenseKeyEx). The advantage of Computer Name bound keys is that a key cannot be resued on other computers (unless they have the same name).
- **User Defined:** User defined keys are computer bound keys. The computer identifier that is used to uniquely identify a computer is user defined. For example, if you want to create a computer bound key that is bound to the serial number of the hard disk, you would need to do the following:

- Create your own function to get the serial number of the hard disk.

- To generate a key manually using the Quick License Manager user interface, you would need to generate the hard disk serial number in your application, display it to the user and ask the user to e-mail it to you. You would then enter this value in the Quick License Manger user interface and generate the key.

- To generate a key programmatically, typically from a web service as shown in the provided sample, you would generate the hard disk serial number in your code and invoke your web service providing the serial number. The web service would then use the CreateLicenseKeyEx API to generate and return a key bound to the serial number of the hard disk.

- To validate a key in your code, you would generate the hard disk serial number and provide that value to the ValidateLicenseEx API.

- **Activation Key:** An activation key is a generic key that does not enable your software. It just allows a user to request a computer bound key. This type of key is typically used for online software registration or activation. This is a typical scenario:

- A customer buys your software online. You automatically generate an activation key and send it to the customer.

- The customer installs your software and enters the activation key. Your software connects over the internet to your web service providing the information required to create a computer bound key. The web service creates the key and returns it to your application.

- Your application then validates the returned computer bound key and enables the software.

-

Features

As of QLM 4.0, you can embed up to 32 features in a license key. Features are divided into 4 sets with 8 features per set.

To define features, click on Define Products, select a product and add your features.

You can generate license keys with embedded features in several ways:

- In the QLM console, click on Generate Keys, select the product and features to enable and then click on Generate.
- Using the QLM CreateLicenseKeyEx4 API (IsLicense50.dll)
- Using the QLM .NET API CreateActivationKey (QlmLicenseLib.dll)
- Using the QLM .NET API CreateActivationKeyWithExpiryDate (QlmLicenseLib.dll)
- Using the QLM .NET API CreateOrder (QlmLicenseLib.dll)

To enable support for 32 features, you need to select the QLM Engine version 4.0.00 or higher.

To verify if a feature is enabled in your code, use the IsFeatureEnabled API. The sample located in: %UserProfile%\My Documents\Quick License Manager\Samples\QLMExpress\DotNet\Basic\C#\vs2005 demonstrates how to verify if a feature is enabled in your code.

Validate license keys in your code

To validate a license key in your code, you need to first add a reference to the Quick License Manager DLLs to your project as described in the table below.

Application Type	QLM Express	QLM Professional or Enterprise
Windows Forms .NET 2.0 or higher	redistrib\.net2.0\QlmLicenseLib.dll redistrib\.net2.0\QlmControls.dll redistrib\.net2.0\QlmControlLicenseWizard.dll	redistrib\.net2.0\QlmLicenseLib.dll redistrib\.net2.0\QlmControls.dll redistrib\.net2.0\QlmControlLicenseWizard.dll QlmLicenseWizard.exe
ASP.NET 2.0 or higher	redistrib\.net2.0\QlmLicenseLibEmb\QlmLicenseLib.dll	redistrib\.net2.0\QlmLicenseLibEmb\QlmLicenseLib.dll
Excel 2003 or higher	redistrib\.net2.0\QlmLicenseLibEmb\QlmLicenseLib.dll or redistrib\x86\IsLicense50.dll	redistrib\.net2.0\QlmLicenseLibEmb\QlmLicenseLib.dll QlmLicenseWizard.exe
MS-Access 2003 or higher	redistrib\.net2.0\QlmLicenseLibEmb\QlmLicenseLib.dll or redistrib\x86\IsLicense50.dll	redistrib\.net2.0\QlmLicenseLibEmb\QlmLicenseLib.dll QlmLicenseWizard.exe
Outlook Add-in	redistrib\.net2.0\QlmLicenseLibEmb\QlmLicenseLib.dll	redistrib\.net2.0\QlmLicenseLibEmb\QlmLicenseLib.dll QlmLicenseWizard.exe
C++	redistrib\.net2.0\QlmLicenseLib.dll or redistrib\x86\IsLicense50.dll redistrib\x64\IsLicense50.dll	redistrib\.net2.0\QlmLicenseLib.dll QlmLicenseWizard.exe
VB6	redistrib\.net2.0\QlmLicenseLib.dll or redistrib\x86\IsLicense50.dll	redistrib\.net2.0\QlmLicenseLib.dll QlmLicenseWizard.exe

What is the different between "QlmLicenseLib.dll" and "QlmLicenseLibEmb\QlmLicenseLib.dll"

QlmLicenseLib.dll includes all the QLM Professional and Enterprise functionality. In addition, QlmLicenseLib.dll acts as a wrapper for IsLicense50.dll which is a C++ DLL. Therefore, QlmLicenseLib.dll depends on IsLicense50.dll. IsLicense50.dll comes in 2 versions: one version of 32 bit systems and another version for 64 bit systems. When you install a .NET application on a 64 bit system, you need to make sure that you install the 64 bit version of IsLicense50.dll. A special version of QlmLicenseLib.dll was created to simplify deployment of your application. The version of QlmLicenseLib.dll located in the QlmLicenseLibEmb folder has the two versions of the IsLicense50.dll embedded as resources. At runtime, IsLicense50.dll is automatically extracted and temporarily stored and then loaded from disk. This approach works well for ASP.NET applications or any application where the user has "write" permissions on the folder when QlmLicenseLib.dll is located.

NOTE: All information relating to an evaluation key (except the key itself which you need to store and retrieve) is stored in the registry under HKEY_CLASSES_ROOT\CLSID\<YourGUID> where <YourGUID> is the unique identifier associated with your product. Quick License Manager handles the case where the system date is set back to bypass the license expiry date. When this occurs, Quick License Manager permanently

disables the evaluation key. The only way to re-enable the license key is to delete the registry entry HKEY_CLASSES_ROOT\CLSID\<YourGUID>.

Protect your software

QLM protects against breaking the licensing in the following ways:

- QLM stores the date the software was first ran. If the user uninstalls the software and reinstalls it, the evaluation period continues from its previous state. This means users cannot just uninstall and reinstall the software to reinitialize the trial period.
- QLM detects if a user sets the date back and disables the license in this event.
- QLM allows you to define computer bound keys so that a license key cannot be reused on another computer.
- QLM provides a mechanism to detect if the QLM DLL was tampered.
- QLM license keys are protected with an encryption that each customer defines, per product. This means that other QLM customers cannot decrypt your license keys.

Protect against tampering of the QLM DLLs

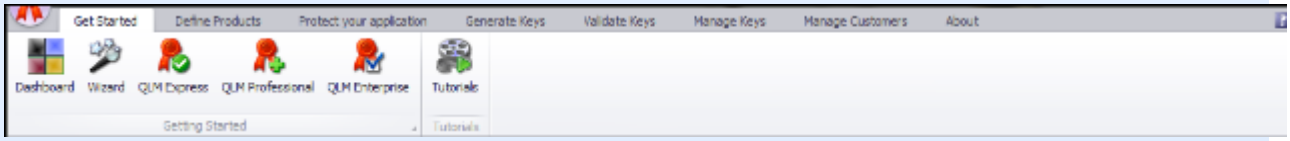
To protect against a user modifying the IsLicense50.dll or replacing this DLL, the IsLicense50.dll is digitall signed with a Microsoft Authenticode Certificate. The QLM .NET API automatically validates the signature of the DLLs prior to loading it.

All other QLM DLLs are .NET assemblies are are digitally signed as well. Any attempt at modifying these DLLs will result in a failure to load the DLL.

User Interface Reference

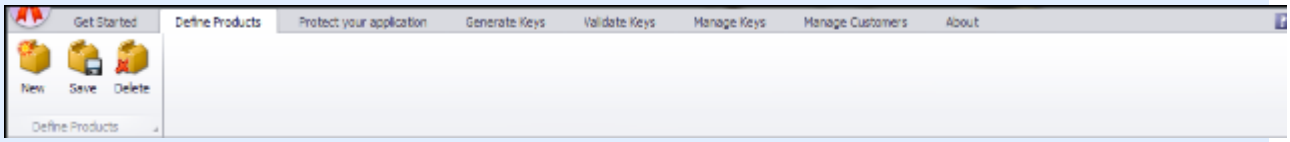
The QLM Console is a comprehensive command center for operating Quick License Manager. Across the top of the Console is a menu bar with commands for accessing each of eight main areas. These are introduced briefly here, then covered fully in the sections that follow.

Get Started



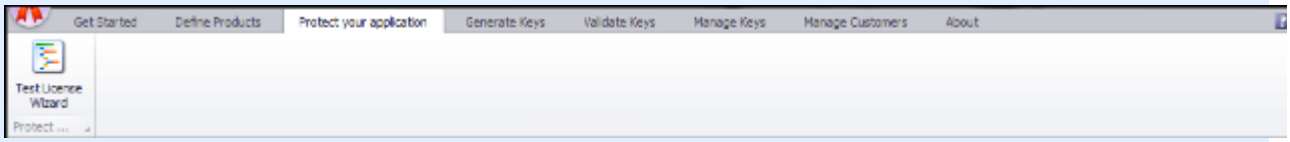
This area is home to the Dashboard, where you'll begin each Console session, and to the QLM Getting Started Wizard, an optional tool for helping you set up the licensing for one of your products efficiently. The wizard will recommend one of QLM Express, QLM Professional and QLM Enterprise as best meeting the licensing requirements of a particular product. It will also provide links to relevant informational and tutorial resources.

Define Products



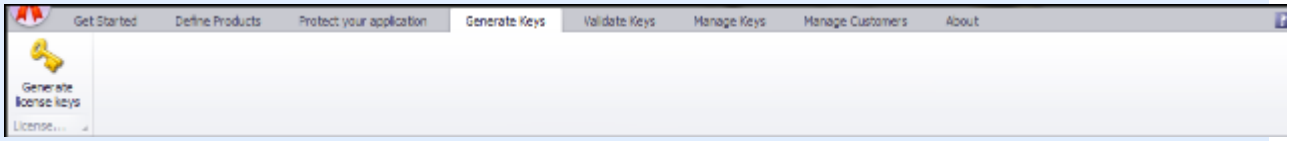
Before you can use QLM to manage the licensing of a product, you must add the product to QLM using the tools in this area. Once a product has been defined, you can begin configuring the keys and licensing options that will govern user access to the product's features.

Protect your application



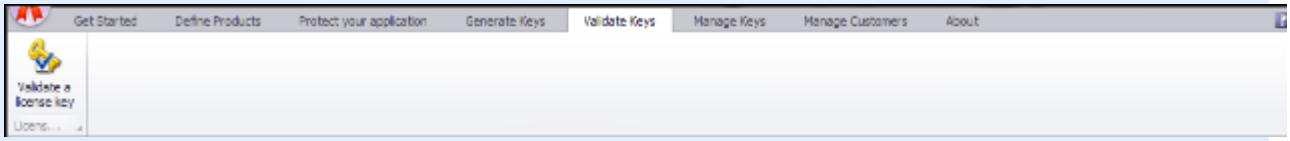
This area contains the Code Generator Wizard, which will generate code in any of several languages for adding license validation to your application. The wizard also lets you customize the built-in license registration form, if you choose to use it, for better integration with your product. To see how your configuration choices for the license registration form will look when it goes live, click the Test License Wizard button in the ribbon.

Generate keys



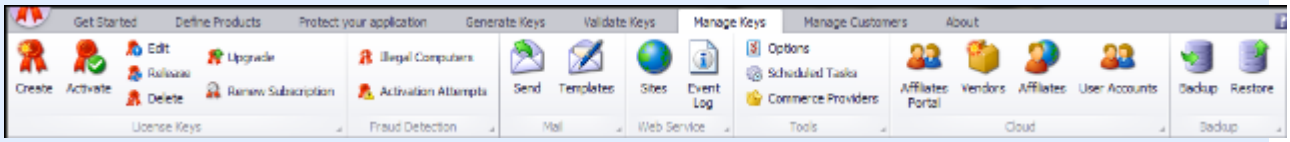
This area's purpose is to generate license keys for your products interactively (rather than in code). Several kinds of keys are supported, including the activation keys that let the user obtain permanent computer-bound or computer-independent keys. Such properties of the key as its expiry terms, and the particular set of features it will enable, can be defined with the controls here. Keys generated from this area are neither saved nor managed by QLM. To generate keys that are published to the database, create keys via the Manage Keys section.

Validate keys



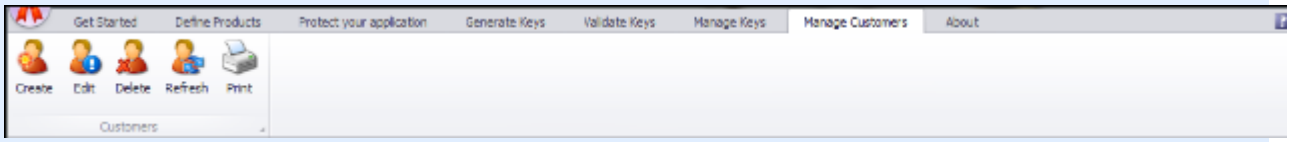
This area lets you check the validity of an existing key with respect to a particular product. If the key is valid, the set of features that it enables, the number of licenses it embeds, and its other properties are displayed.

Manage keys



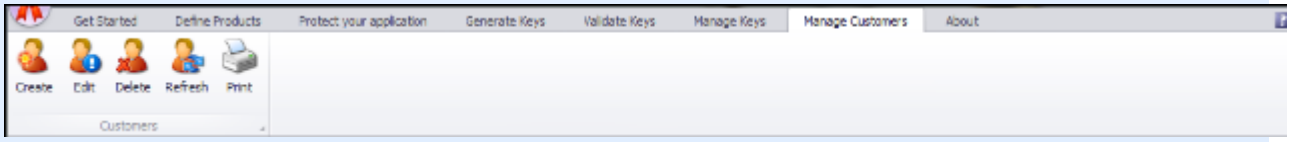
With its many controls, this area provides access to more than 20 license management functions in seven control groups: License Keys, Fraud Detection, Mail, Web Service, Tools, Cloud and Backup. The tools and reports available here make Manage keys a particularly 'key' area of the Console's user interface.

Manage customers



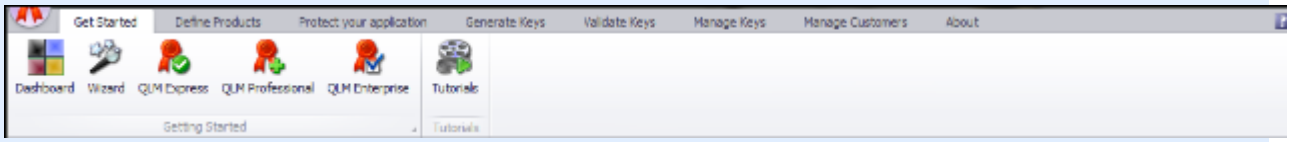
The customer records in your QLM database are the focus here. Controls are provided to create, edit, print and delete records, and to refresh synchronization with the database.

About



This area displays your licensing information for this copy of QLM. If you have not yet activated your license, you can do so here, using the same QLM license activation form provided as a programming component to QLM customers. Another control in this area checks for program updates; this too demonstrates the use of a QLM capability available to customers.

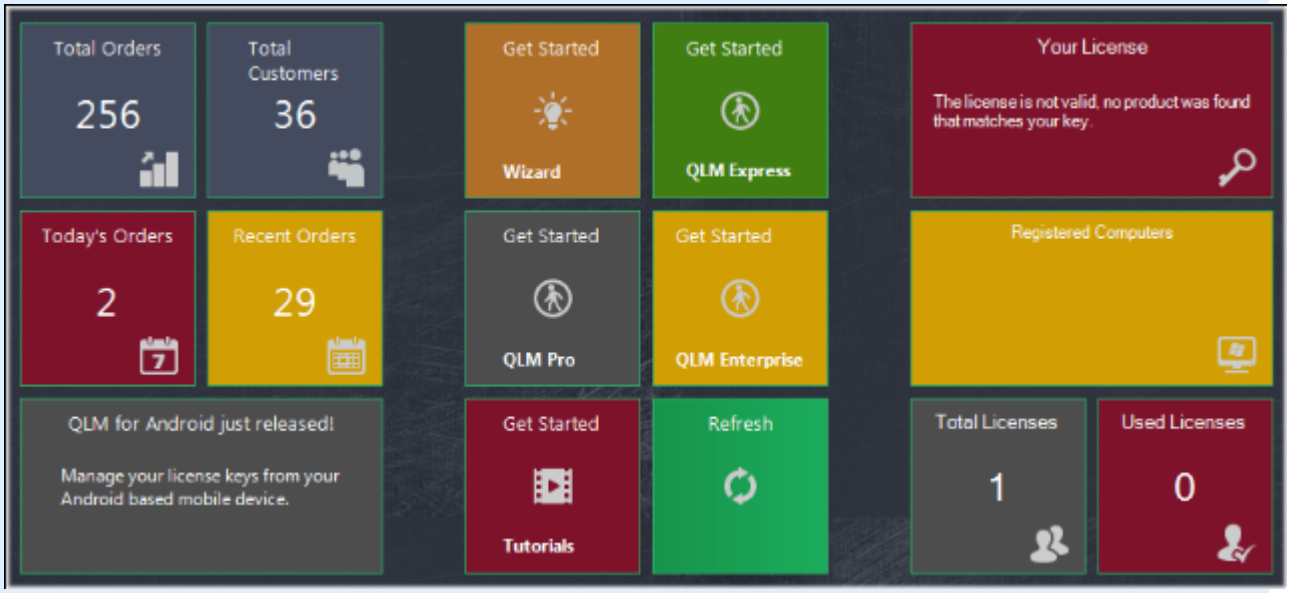
Get Started



The first of the QLM Console's eight major functional areas is accessed with Get Started at the left-hand end of the Console's main ribbon bar. Two Console features with which you will quickly become familiar are the Dashboard, which is the starting point of each session, and the Getting Started Wizard, an optional but handy tool for helping you set up product licensing under QLM. Both of these are found in the Getting Started control group, along with information pages detailing each of the three levels of QLM: Express, Professional and Enterprise. The single button in the Tutorials control group provides a link to online information resources with detailed information on selected topics.

THE DASHBOARD

The Metro-style tiles that make up the Dashboard are organized by default into three columns. A row in any column contains either one full-width or two half-width tiles. (If the window height is sufficiently reduced, doubled columns or tripled columns may be seen.) The organization of the Dashboard can be customized by dragging individual tiles from one column to another, or into an empty location to create a new column. As you drag a tile to locations in other columns, the tiles already there move aside as necessary to indicate where the new arrival will lodge. The default layout of Dashboard tiles in three columns.



System Information

The four smaller tiles in the left-hand column of the default layout show vital statistics from the QLM database, with counts of customers, total orders, today's orders, and recent orders. (See Manage Keys ¬ Options to review or adjust the meaning of 'recent'.) The bottom tile displays news items about QLM. Click an item to view the full story in your web browser.

Quick License Manager Information

The tiles in the center column provide information about the latest QLM version as well as News Releases about QLM. The Refresh tile resynchronizes the Dashboard display with the QLM database.

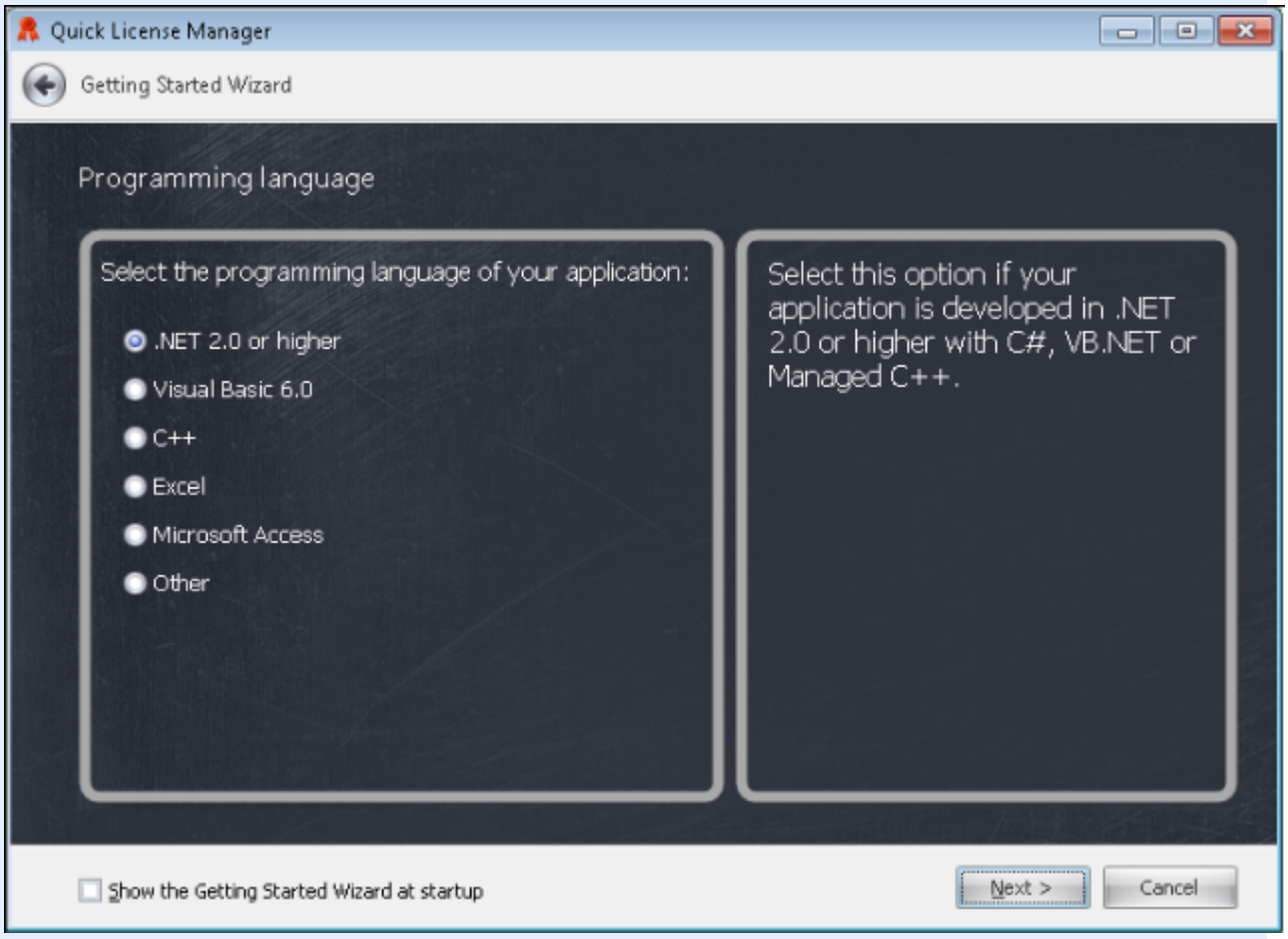
License Information

The tiles in the right-hand column of the default Dashboard layout show information about your QLM license. The upper box gives a verbal status report regarding the license. The one beneath it lists the computers on which you are registered to run QLM. In the bottom row are displayed the number of licenses you have available, and the number of those already in use.

GET STARTED WIZARD

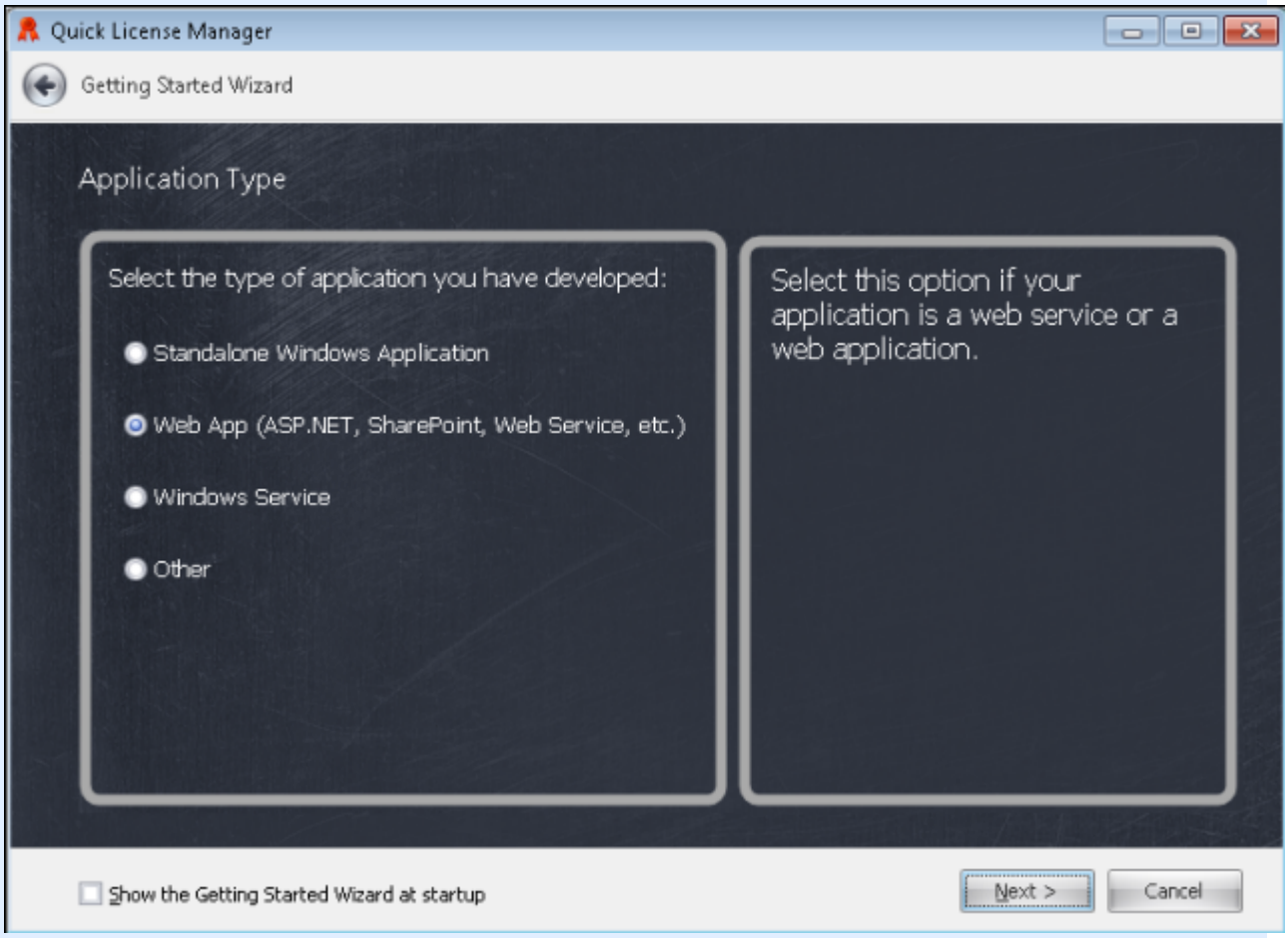
The Getting Started Wizard helps you find the best path into QLM for a particular licensing scenario by leading you through a questionnaire to determine three requirements:

Programming language: .NET 2.0 and higher languages are directly supported by QLM, as are Visual Basic 6.0, C++, Excel and Access. Other languages can also make use of the QLM API if they are capable of interfacing with .NET assemblies, COM objects (ActiveX), or native DLLs.

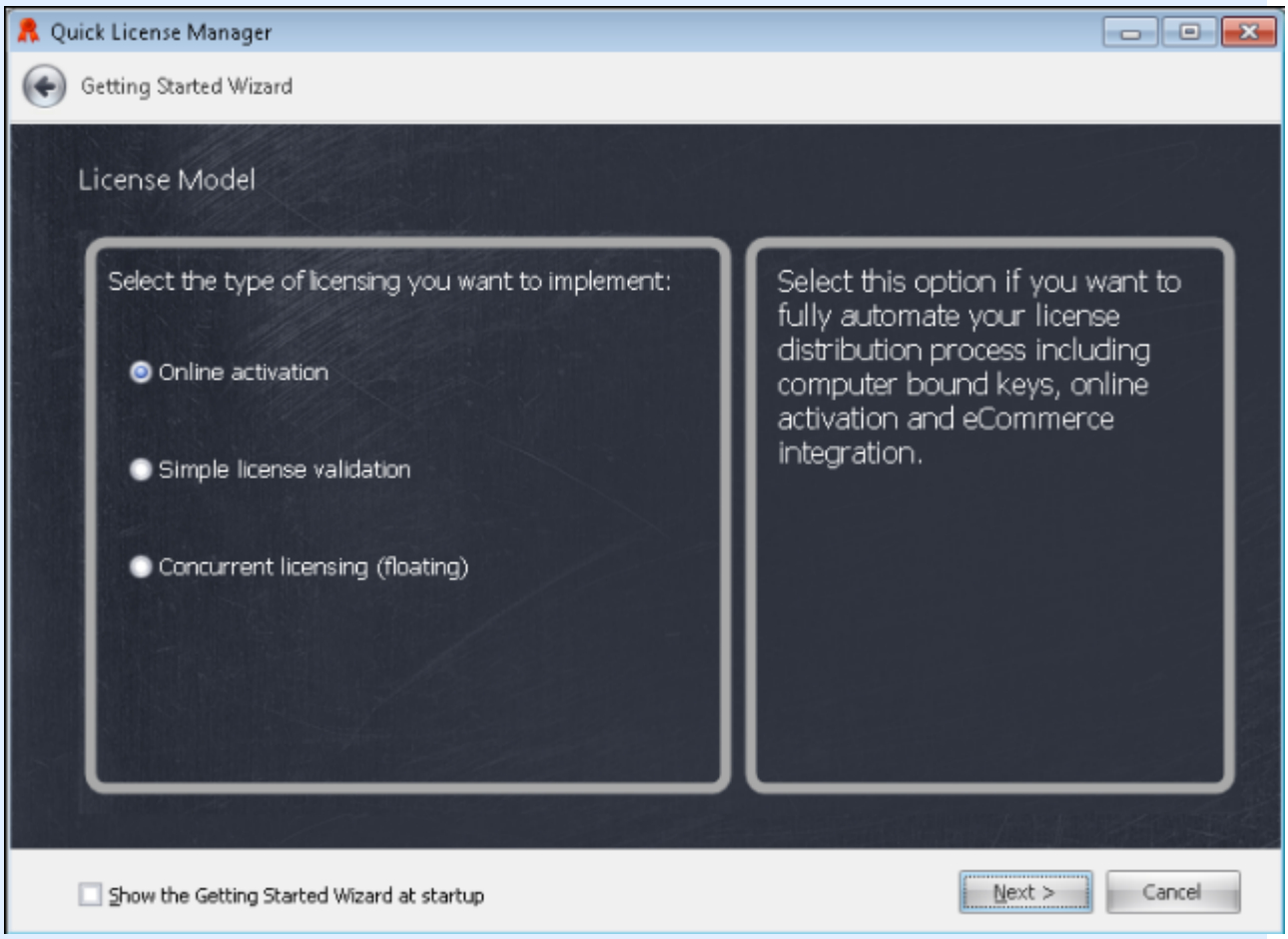


Application type: The wizard offers "Standalone Windows application", "Web App", and "Windows Service" as configured choices. If your application does not fall into one of these categories ("Other"), contact us for advice

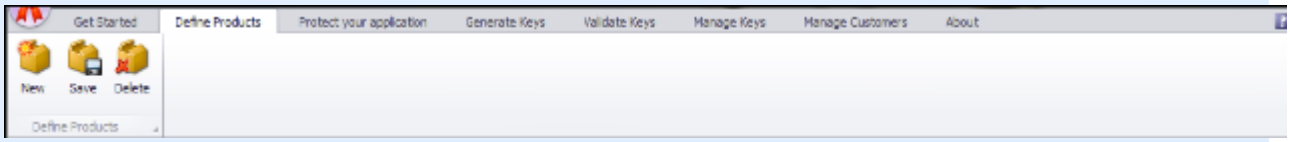
specific to your situation.



License model: Your answer to this question determines the edition of QLM that you will need for your application. For instance, if you select "Simple license validation" as your license model, the wizard will recommend QLM Express as best meeting your needs. The wizard then provides a page of links to relevant tutorials and code samples that will help get you started without wasted effort.


Quick License Manager
Getting Started Wizard
License Model
Select the type of licensing you want to implement:
Online activation
Simple license validation
Concurrent licensing (floating)
Select this option if you want to fully automate your license distribution process including computer bound keys, online activation and eCommerce integration.
Show the Getting Started Wizard at startup
Next >
Cancel

Define Products



A QLM license key is associated with a particular product as identified by its name and version in the Define Products tab.
To add a product, click on the Add button and enter the product information.
The full set of properties is divided amongst four tabs, *Product Information*, *Latest Version*, *Keys* and *Vendor*.

PRODUCT INFORMATION

Product name: Enter the base name of your product without versioning.

Major version, Minor version: This pair of numbers, both of which can have at most two digits, specify a particular generation and release of the product for licensing. Taken together, the name and the two version fields constitute a unique handle by which this product will be identified in QLM.

Encryption key: Enter an encryption key to encrypt the license key. An encryption key is like a password. A license key can only be decrypted with this password. The ! character is not allowed in the encryption key. Encryption keys are no longer used as of QLM 5. A PKI algorithm is now employed for encrypting the license key. Each time you define a product in QLM, a key pair is created for it. Typically a different key pair is used for each product. You will use the private key in the pair to generate license keys. The public key is used to decrypt license keys. It must be included in your code and set before licenses can be validated.

Product ID: This field is automatically generated when the database record for the product is stored, and remains permanently associated with it from then on.

Release Date: The Release Date is used in conjunction with the Maintenance Plan feature to determine upgrade entitlement. Provided the Maintenance Plan Renewal Date is greater than the Release Date, the customer is entitled to the upgrade.

GUID: This identifier is automatically generated for your project by QLM. It is used as a key to the evaluation information for the product in Windows Registry under: HKEY_CLASSES_ROOT\CLSID\<GUID>. The New button adjoining the field generates a new GUID. Calls to the DefineProduct API in your application will need to be updated if you do this.

Features: Up to four feature sets with up to eight licensable features in each can be defined for your product. The licenses for the product can be configured on creation to enable one or more of these sets, thus making their individual features available to the user who holds the license.
The information about the features you have so far defined for the product is found on the Features list. Each line displays the set number (0 to 3), a numeric ID, and a name provided by you. The numeric values used as IDs are those corresponding to the individual bits of a flag byte: 1, 2, 4, and so on up to 128. To manage the Features list, three buttons are provided: Add, to define a new feature; Edit, to change the assignable properties of the selected feature – its name and the set that it belongs to; and Remove to discard the feature altogether.

LATEST VERSION

The latest version tab has three fields that let you implement a "Check for Updates" feature in your software.

Latest Version: Enter in this field the latest version of your product. The format of this value is up to you. The availability of a particular update will be determined in your running software by retrieving this value and comparing it with the current version.

URL to latest version: Enter a URL to the latest version of your software for the QLM updating framework to use.

Notes about latest version: The notes you enter in this text box can be retrieved for display to the user along with other information about the update.

KEYS

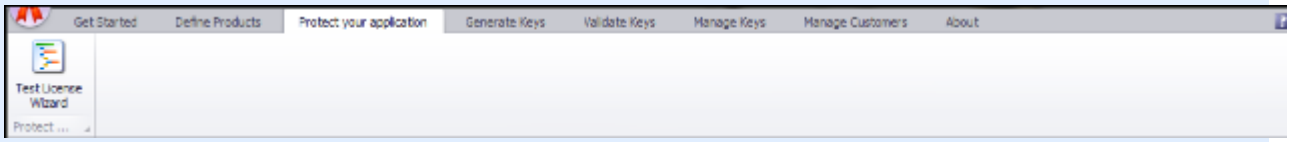
QLM uses asymmetric encryption to encrypt license keys. The algorithm requires that a pair of encryption keys, one public and one private, be predefined. QLM generates such a pair for you automatically when your product is created. By default the same key-pair is used across all versions of the same product.

Public key: You will need this value in order to validate a license in your program code.
Private key: This value is used for generating licenses. For the security of your licensing process, it is recommended not to generate licenses in the protected software itself, but to use an external mechanism for that purpose.
New: Click this button to generate a new key pair at any time. Remember, however, that your software must be updated to use the new public key value.
Unmask: Check this box to view the actual text of your private key rather a string of asterisks.

VENDOR

The Cloud Edition of QLM allows you to associate a product with a particular vendor, by selecting the vendor from the dropdown list and entering the vendor's own unique ID for the product.

Protect your application



QLM provides several approaches to protect your application. The Protect your application wizard generates a helper class to include in your application as well as 2 configuration files that specify the properties of the QLM License Wizard Control. The QLM License Wizard Control is a license registration form to capture a license key and activate it.

To configure the Wizard, the opening page asks you to make three selections. First, a dropdown list under Select the product you want to protect lets you choose any product you have configured in the Define Products area. Second, another dropdown list under Select the web service that manages this product offers any web services profiles you have defined in the Web Services Settings dialog (Manage Keys / Sites). If you are using QLM Express, you can simply pick the Default profile.

Finally, in a box under Select the programming language of your application, check the radio button that best describes the type of your application. The available options are:

- .NET Framework 2.0 or higher: When this is selected, two subsidiary options – C# and VB.NET – become available.
- VB6, VBA, ActiveX/COM capable scripting language: Choose this option if your application is designed to run within a VB6 or Microsoft Office application like Excel or Access.
- VC++ 6.0 and higher: This choice designates a native MS Windows application written in Visual C++.

If your application is written in any other language, contact us for guidance on how to integrate QLM to your application.

The second page of the **Protect your application** wizard lets you configure the look and feel of the QLM License Registration Wizard. Changes you make to the properties in the left-hand panel are reflected at once in the live preview occupying the rest of the page.
The final page of the wizard allows you to customize the QLM license objet used by the LicenseValidator helper class.

PROTECTING .NET APPLICATIONS

For Windows Forms .NET applications, QLM provides 3 .NET Controls that you can easily drop in your application. These controls are forms that allow the end-user to enter a license key and activate it.
The 3 .NET controls are:

- QLM Express: QlmExpressLicenseValidationControl (QlmControls.dll)
- QLM Pro/Enterprise: QlmWebBasicActivationControl (QlmControls.dll)
- QLM Pro/Enterprise: QlmLicenseWizardCtrl (QlmControlLicenseWizard.dll).

Both QLM Pro/enterprise controls offer very similar functionality. The main difference between these 2 controls is that the QlmLicenseWizardCtrl uses a wizard based graphical user interface. In addition, the QlmLicenseWizardCtrl can read its properties from 2 external configuration file. These configuration files are generated by the **Protect your application** wizard.

The QLM License Wizard Control is also available as a standalone executable that can run alongside your application.

When you install QLM on your system, a Quick License Manager tab is added to your Visual Studio toolbox that contains the QLM .NET Controls. If for any reason the Quick License Manager tab was not added to your Visual Studio toolbox, you can attempt to recreate this section by clicking on the Refresh button under Options / Enable Visual Studio Integration. Note that the QLM tab is not added to the Visual Studio Express edition as this edition does not support programmatic additions to its toolbox.

For more details about the QLM .NET Controls, refer to the API Reference.

For WPF applications, you can host the QLM Windows Forms Controls in WPF as described in this article.

If you are developing a web based **ASP.NET** application, the above .NET controls cannot be used. A sample program is available that shows how to capture and activate a license key. The sample is located here: *%userprofile%\Documents\Quick License Manager\Samples\qlmpro\DotNet\Basic\C#\vs2008\AspDotNetSample*

In addition to the .NET Controls that are required for capturing and activating a license key, the **Protect your application** wizard generates a helper class that you need to add to your application. The helper class has a method called **ValidateLicenseAtStartup.** You should call this method when your application is launched. For more details about this method, refer to the API reference section.

PROTECTING NON .NET APPLICATIONS

For non .NET applications, you have 2 options to catpure and activate a license: (a) you can use the standalone *QlmLicenseWizard.exe* application to capture and activate a license or (b) you can create your own license registration form. The former is clearly the simplest approach.

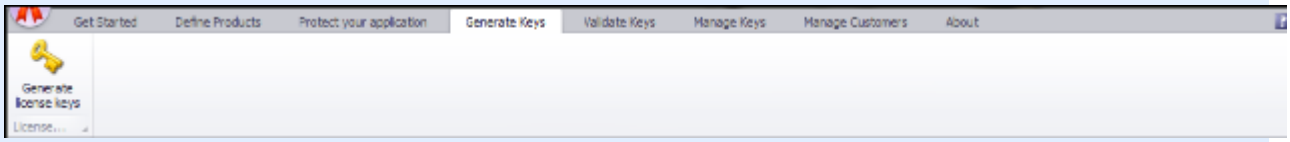
For VB or C++ applications, the C++ applications, the **Protect your application** wizard generates a helper class that you need to add to your application. The helper class has a method called ValidateLicenAtStartup. You should call this method when your application is launched. If the call to **ValidateLicenseAtStartup** fails or returns that activation is needed, you should then invoke the *QlmLicenseWizard.exe* as follows:

QlmLicenseWizard.exe /settings "<path>\settings.xml" /uiSettings "<path>\uiSettings.xml"

where both xml files referenced above are generated by the **Protect your application** wizard.

When the *QlmLicenseWizard.exe* application exits, you should once again call **ValdiateLicenseAtStartup** and confirm that the license is valid. If it is not, you either exit your application or launch the *QlmLicenseWizard.exe* application again.

Generate Keys



The **Generate Keys** tab of the QLM Console lets you configure and create a license key – or a list of license keys – for any of your products.
Note that license keys generated from this tab are not stored in the **QLM** database. To create license keys that can be used for online activation, refer to the *Create* option under the **Manage Keys** tab.
Select your product on the *Product Name* dropdown, then continue with the other controls in the Options box as follows:

Number of embedded licenses: Enter here the number of licenses to embed in each generated key. For example, if you enter "2" the user holding one of these keys will be permitted to activate it on two different computers.

Number of license keys to generate: Enter the number of license keys to generate when you click the *Generate license keys* button. This options is designed for working with online payment systems that issue the next available key from a prepared list when your software is purchased. Note that licenses distributed in this way do not benefit from QLM's database integration.

Engine version: Specify the version of the QLM Engine to use. If your application has shipped with an earlier version of QLM, select the version of the QLM Engine that your application is using. Note that the latest version of the QLM Engine is 5.0.00.

Permanent License: Check this option if you want the generated license keys to remain in effect permanently once issued.

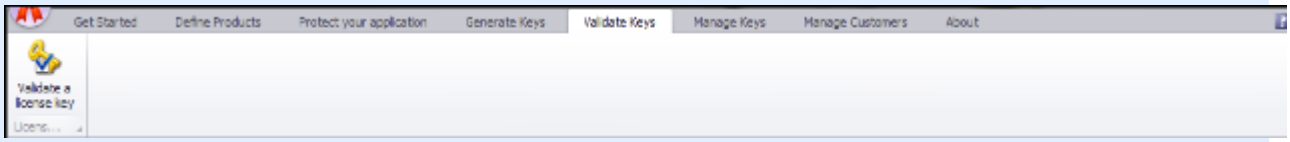
Expiry Criteria: Check this option for trial or subscription based licenses that should expire within a certain number of days of installation, or upon a predetermined date.

Features: Check the boxes next to the features that you want this license to enable. The list of available features can be configured in the *Define Products* tab.

License Type: This dropdown selects which of the four available QLM license types will be generated. The "Bound to Computer Name" and "User Defined" types both request additional information in the *Computer Identifier* field.

When you have completed your license settings, click the *Generate license keys* button in the ribbon strip. The key, or set of keys, you have requested appears in the *License Keys* box. Click the *copy* button to the right of the box to transfer the information to the Windows Clipboard. The final destination – a customer email, a configuration page on-line, or elsewhere – is up to you.

Validate Keys



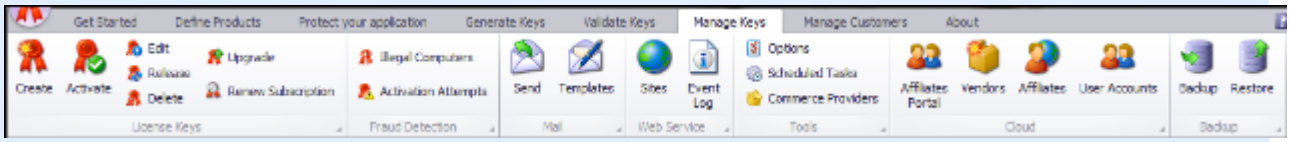
The **Validate Keys** area of the QLM Console allows you to interactively test a key to determine whether it is valid.

To validate To validate a license key, paste it into the *License Key* text field and click **Validate a license key.**

If the license is valid, the information contained in the key is decrypted and displayed in the form.

IMPORTANT: For computer bound license keys, you need to enter the *Computer Identifier* field prior to validating the license. QLM cannot validate a computer bound license key without the *Computer Identifier*.

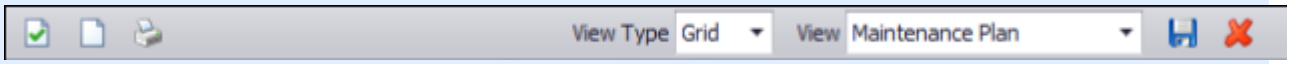
Manage keys



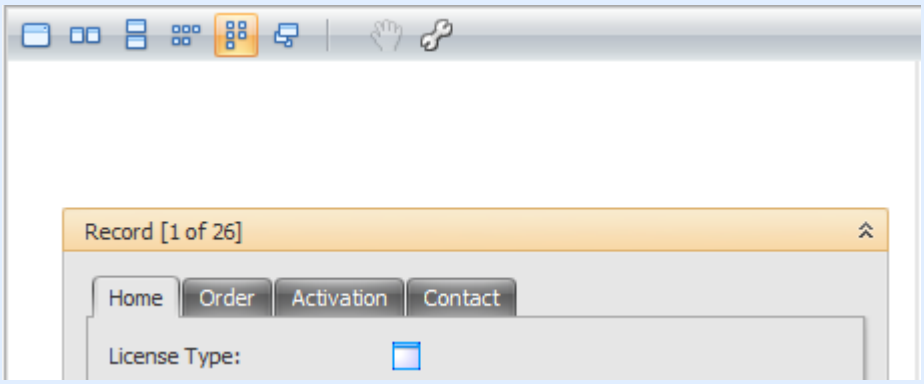
The Manage Keys area in the QLM Console provides a task-oriented interface to the QLM database located on your web server. Communication between the Manage Keys section and the database occurs via the QLM web service which must be configured from the Sites button.

The main window permits interactive configuration of the selection, ordering and grouping of result columns. A query builder across the the top of the display lets you access data according to whatever combinations of criteria you specify.

On the right, a column of preset queries provides answers to often-asked questions such as, "How many licenses have we sold today?", "How many activation licenses were obtained yesterday for our products?", and "Which customers on maintenance licenses are up for renewal in the near future?".



Items are shown in a grid or card view. To switch between the grid view and the card view, click on the View Type dropdown box in the status bar and select the view of your choice. Views can be customized and saved. To save a view, click on the Save icon in the status bar. To customize the fields displayed in the card view, click on the wrench button on the card view toolbar.



LICENSE KEYS

The buttons in this control group allows you to create or manage licenses. Note that all operations described in this section can also be automated via the QLM API. Some operations are also available via regular http requests that can be invoked directly from an eCommerce provider.

CREATE

This button opens the Create License dialog window for creating activation keys interactively and publishing them to the QLM database. Providing such a key to a customer at the time of sale begins a licensing transaction that ends when the customer successful enters the activation key into your installed application and a computer bound license key is generated and sent back to the customer.

ACTIVATE

Activates an activation key and generates a computer bound license key. Use this button to activate one or more keys interactively for testing purposes or as other circumstances may require. Ordinarily, activation should be performed by your software while running on the customer's machine. To activate a key, you need to specify a unique computer identifier for the customer's computer. The Computer Name field is optional but recommended.

EDIT

This button brings up the Edit License Information dialog window, which provides controls for editing the data associated with a license.

RELEASE

Customers may sometimes require a transfer of their license key to another computer. The Release button clears the existing association between this license and a computer; once this has been done, the license can be reactivated on the new system. QLM maintains a history of all released licenses that can be queried by means of the **Search Released Licenses** item in the Search dropdown at the top right of the area header.

DELETE

This button deletes, after confirmation, one or more licenses selected in the data browser. Deleting a license key removes all records associated with the key.

UPGRADE

Upgrade your customer to a new version of your product, an additional set of features, or a new version of the QLM Engine. The upgrade process generates a new activation key, which you must convey to the customer for use. The previous activation key is archived.

RENEW SUBSCRIPTION

If you sell your software as a subscription, you can renew subscriptions without sending your customers a new license key by moving back the subscription expiry date using the Renew Subscription date window. Customers simply reactivate their existing license keys to renew their subscriptions. When the subscription associated with a license is extended, a computer-bound key is generated with the revised expiry date. The customer receives the new key by activating the license.

CREATE

The Create button creates an activation key. Activation keys can also be created via the QLM API and more typically via QLM's integration with eCommerce providers. For more information about how to automatically create activation keys during the purchasing process, read the eCommerce Providers section in the QLM Professional help.

- **Product**: The product associated with the license key.
- **Number of keys**: The number of license keys to generate.
- **Number of activations per key**: The number of activations that this key will allow. QLM manages the number of activated licenses and prevents the user from activating more than the number of purchased licenses. For example, if a customer purchases 5 copies of your software, rather than sending the customer 5 license keys, you create a sngle license key that can be activated on 5 different computers.
- **Floating seats**: To implement Floating Licenses (requires QLM Enterprise), specify the number of floating seats associated with this activation key.

  You can alternatively use this field to control the number of instances of your application that can run in a Terminal Server session. An instance is uniquely identified by a Terminal Server ssession for a specific user. To enable this feature, set the LimitTerminalServerInstances property of the QlmLicense object to True in your code.
- **Customer E-mail**: Associate the key to a customer.
- **Features**: Check any feature to enable in this license key (if you have not defined any features for your product, this field will be empty).
- **Affiliate**: Select from this dropdown list the affiliate that sold the license, if applicable.
- **Generic License**: When this option is checked, the license key to be generated upon activation will be generic – not bound to any particular computer.
- **Maintenance Plan**: When selected, license keys becomes version-agnostic. A customer running an older version of your product will be able to activate his license key with the latest version of your product. Use this option if you have offered your customers free upgrades to all future versions or if the customer has purchased a maintenance plan. For more details, read the Maintenance Plan section in the QLM Professional help.
- **Expiry Criteria**: If you would like the license to be time-limited, check this box to enable its subordinate controls.
- **Engine Version**: If you have customers using an older version of QLM, select the appropriate version of the QLM engine from this dropdown list on the Advanced tab. It is highly recommeded to use QLM Engine version 5.0 or higher.
- **User data**: Use this text area on the Advaned tab to record any notes or text-encoded data to associate with this license.
- **Comments**: Use this text area on the Comments tab to record any notes or text-encoded data to associate with this license.

Finally, the Save Defaults button allows you to store default values for all the fields above. Subsequently launching the Create dialog will automatically load the saved default values.

ACTIVATE

The Activate button manually activates the selected license. Note that activation typically occurs from within your application via the QLM API. Manual activation through the QLM Console is required in a situation where the end-user cannot activate a license due to the lack of an internet connection. This is typically referred to as Manual Activation v/s Online Activation.

- **Activation Key**: The key to activate (this field is read-only).
- **Customer E-mail**: The customer's email address from an existing customer record.
- **Computer Identifier**: A unique identifier for the customer's computer. This can be the computer name, the MAC address, the hard disk serial number or any other identifier of your choice.
- **Computer Name**: The name of the computer. Setting this field is not required but is recommended for easy identification of the customer's system.
- **Product**: The product associated with the license key.
- **Engine Version**: The version of the QLM Engine. If you have customers using an older version of QLM, select the appropriate version of the QLM engine.
- **Affiliate**: The version of the QLM Engine. If you have customers using an older version of QLM, select the appropriate version of the QLM engine.
- **User Data**: Associate any data to the license key

FRAUD DETECTION

QLM can track illegal computers that connect to the web service and logs information about these computers in the database. An illegal computer is defined as a computer that has a valid license key but whose license key is (a) not in the database or (b) in the database but registered for another computer. This situation can occur if a user with a valid license key requests that his license be released from computer A claiming to have uninstalled your program from computer A. If the user subsequently attempts to connect to the web service via computer A, QLM detects this computer as an illegal computer and logs it in the database.

ILLEGAL COMPUTERS

This button displays a list of all the detected illegal computers. QLM does not prevent all illegal computer from running your application. Once an illegal computer is detected, it is up to you to decide the course of action to take by contacting the customer and enquiring about the situation.

ACTIVATION ATTEMPTS

Click the Activation attempts button to view a list of failed activation attempts. A failed activation is typically due to multiple attempts to activate the same license key on different computers. Though failed activation attempts may indicate an intention to illegally activate a license on a particular computer, they can also be regarded as the sign of a user who needs additional licenses for your software.

MAIL

QLM provides a tool to send personalized emails to your customers. Note that QLM sends emails via your Outlook client.

SEND

To send a personalized email to a set of customers:

- Perform a search that returns a set of data.
- Select the items that you want to send an email to or click on the Select All button.
- Click on the Mail/Send button.
- Select the Outlook profile to use.
- Select the email template to use.
- Select the E-mail account to use.
- The **To** field should already contains the email of the selected customers.
- The additional **To** field allows you to type additional email addresses. Note that if your email message uses variables, these variables will not be expanded when sending emails to the additional recipients.
- Enter the Subject of the email.
- Enter the body of the email.
- Click on the Send button.

To create a personalized message, you can use variables within the **Subject** and the **Body** of the message. Any of the visible columns in License Management tab may be used as variables.
For example, your message could read:

```
Hi %FullName%,
Thank you for your recent purchase of %ProductName%
%MajorVersion%.%MinorVersion%.
Your license key is: %ActivationKey%.
Regards,
Tom
```

TEMPLATES

Templates allow you to create common email content that can be readily used when sending manual or automated emails to your customers. Automated emails can be configured from the Scheduled Tasks option.

WEB SERVICE

The options in this control group pertain to the QLM Web Service, which acts as an interface between a client system and the QLM database. Although you will typically have a single web service, the QLM Console can manage a list of web service profiles, each with its own server address along with data such as the type of database engine on the server, the encryption keys for communicating with the web service and so on. Until you set up your own web service, and during your QLM trial period, you can use the Default web service provided by Soraco. Note that the Default web service does not allow you to upload your own products and can only manage keys for the built-in Demo product that ships with the trial version of QLM.

SITES

Click this button to open the Web Services Settings dialog window, which lets you create, edit and select your connection to the QLM Web Service.

- **URL**: The URL to the QLM Web service is typically of this form: http://yourdomain/yourvirtualdirectory/qlmservice.asmx
- **Authentication Method**: Anonymous | Forms Authentication | Windows Authentication. In most cases, you will want to configure the web service for Anonymous Authentication.
- **Database**: Select whether the database you installed on the server uses Microsoft Access or Microsoft SQL Server.
- **Database Schema**: When a new version of QLM is released, the QLM database schema may require updating. This button verifies and updates your DB schema as required.
- **Path to products file**: The products you define in QLM are initially stored locally on your computer. Once you are satisfied with your products definition, your products need to be uploaded to the QLM database. QLM automatically detects if your products are out of sync with the server and prompts you to either upload or download products from the server. If you want to force an update, click on the appropriate button in this section. Use this feature with caution as you may inadvertently overwrite the QLM database with the wrong products or vice versa.
- **Communication/Admin Encryption Key**: Communication between a client and the QLM web service is protected via an encryption mechanism that prevents hackers from directly calling your web service. This is critical due to the fact that the web service is typically configured to allow anonymous connections.
- **Test**: Once you have configured all fields, click on the Test button to validate that you can properly connect to the QLM web service.

EVENT LOG

Errors detected in the QLM web service are stored in the QLM database. Use this option to view these errors. At times, you may want to increase the verbosity of the messages that the QLM web service logs. This can be accomplished by updating the **loggingLevel** setting in the web.config file of the QLM web service. The highest logging level is 15. The recommended value is 3.

If you increase the **loggingLevel** to 15 to diagnose a specific issue, remember to set it back to 3 to avoid bloating of your QLM database.

TOOLS

The buttons in this control group pertain to miscellenaous options such as QLM console configuration, Affiliates Portal configuration, Dashboard settings and eCommerce Provider options.

OPTIONS

- **Path to searches files** : Searches configured in the right hand panel are stored in a file called queries.xml. You can configure the path of this file. This is typically used if you want to share searches with different people on your team and store searches on a shared path (UNC) on your LAN.
- **Display options**: When you create a query on the fly by selecting the "field", "operator" and "value" in the toolbar over the data set grid, QLM can display the details of your query in the status bar as you build it.
- **Recent Orders**: Allows you to configure the horizon of the recent orders displayed in the QLM dashboard.
- **Upcoming Maintenance Renewals**: Allows you to configure the horizon of the upcoming maintenance renewals displayed in the QLM dashboard.
- **URL to Affiliates Portal**: Allows you to configure the URL to the Affiliates Portal.
- **QLM Agent**: Allows you to configure whether the QLM Agent automatically starts when you login. The QLM Agent is responsible for running the QLM Scheduled Tasks.

SCHEDULED TASKS

Click the Scheduled Tasks button to open a dialog window where you can define operations to be carried out automatically. The scheduled tasks that QLM can execute include emailing notifications to your customers, and displaying alerts when the QLM database is updated. QLM installs with several scheduled tasks predefined and ready for use. One task is designed to handle the email notifications sent to maintenance plan customers; the others are of the alert type. You can use the supplied tasks as-is or customize them to suit your needs. You can also define additional scheduled tasks if required.

Note that as you transition from a trial version of QLM to a purchased version with your own web service, you should modify the scheduled tasks to point to your own web service instead of the Default/Demo web service.

COMMERCE PROVIDERS

The Commerce Providers button opens the Edit Commerce Providers dialog widow, which lets you quickly configure integration between your instance of the QLM Web Service and any of the major online commerce providers. The typical settings that you will need to modify are the User / Password. For more details, review the eCommerce Provider section under QLM Professional.

CLOUD

The Cloud edition of QLM provides several features such as:

- The ability to manage products from different software vendors.
- The ability to provide those software vendors with user accounts that allow them to manage their license keys from the QLM Console.
- A web portal for your affiliates to generate license keys.
- An authentication framework for limiting access to the web portal.

AFFILIATE PORTAL

The Affiliates portal button provides a shortcut to access the affiliates portal. To configure the URL of the Affiliates portal, click on Tools / Options button.

AFFILIATES

If you sell your product via affiliates, QLM allows you to define affiliates and then allow these affiliates to login to the Affiliates Portal for the purpose of generating license keys for their customers. You can control how many trial or permanent license keys an affiliate can create.

To provide an affiliate access to the Affiliates Portal, create a user account for that affiliate as described in the section below.

For more details about affiliates and vendors, review the corresponding help sections under QLM Professional.

USER ACCOUNTS

User accounts are used for 2 purposes:

- Provide access to the Affiliates Portal
- Provide limited access to a QLM Console when the web is service configured with **Windows Forms authentication**.

A user account can be associated to an affiliate, a vendor or both.

When a user logs to the QLM console with an account associated to a vendor, that user will only see license keys for products associated to that vendor.

When a user logs to the Affiliates Portal with an account associated to an affiliate, that user will only see license keys associated to that affiliate.

VENDORS

If you manage software products from different Vendors, QLM allows you to define your vendors and provide each vendor a user account to connect to the QML web service. A vendor can then install the QLM Console on their system and connect to the web service using the provided account. Since the account is tied to a specific vendor, the vendor only sees license keys related to their product. Note that each product needs to be associated to a specific vendor from the **Define Products tab / Vendors**.

BACKUP

QLM can schedule and perform backups of your database to your local computer. No special software is required on the server side to perform the backup or the restore.

BACKUP

To create a backup job:

- Click on the Manage Keys tab.
- Click on the Backup button.
- Click on Add.
- Select the Enable checkbox.
- Type the name of the backup job.
- If you have multiple web servers, select the web service profile to use. Otherwise, select the Default profile.
- Specify how often the backup job will run.

RESTORE

To restore a backup job:

- Click on the Restore button.
- On the left pane, expand the backup job and select the snapshot to restore.
- On the right pane, select the tables to restore. Note that existing data will be deleted.
- Click on the Restore button.

MANAGE CUSTOMERS

The Manage Customers area of the QLM Console provides five functions for managing customers' records in the QLM database.

CREATE

Create a new customer in the QLM database. Customers are uniquely identified by their email address. If you define 2 customers with the same email address, the records of the 2 customers will be merged.

EDIT

Edit the information of the selected customer.

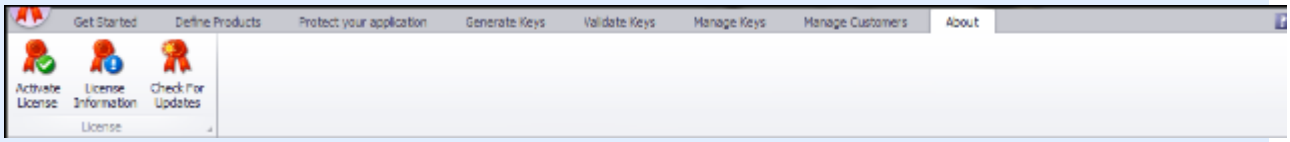
DELETE

Delete the record of the selected customer.

REFRESH

Refresh the list of customers.

About



This tab displays licensing information about your QLM copy.

ACTIVATE LICENSE

When you purchase QLM, you receive an email with a license key that typically starts with the letter 'A'. To activate your purchased copy of QLM, enter the activation key and click on the Activate button.
The activation process connects over the internet to the Soraco QLM web service and activates your license. If you connect to the internet via a proxy server, click on the Proxy Settings button to configure your proxy server.

LICENSE INFORMATION

This section displays information about your QLM license.

CHECK FOR UPDATES

This section displays version information about your QLM copy and provides a link to download the latest version.

QLM Express License Validation .NET Control

The QLM Express License Validation .NET Control is a Windows Forms based .NET control that you can easily add to your Windows Forms application to capture and validate a license key.

The control can be found in the Visual Studio toolbox, in the Quick License Manager section.

To integrate QLM Express with your application, you also need the automatically generated helper class. This class can be generated from the Protect your application wizard.

Following is a list of all the properties that can be set on the QLM Express License Validation .NET Control.

Name	Description
QlmCloseButtonVisible	Show or hide to Close button
QlmComputerID	Set the computer ID to use when activating the license. This property should be typically set programmatically at runtime.
QlmEncryptionKey	Set the encryption key. This property is only required when using QLM engine version 4.0 and earlier.
QlmEvaluationHaveKeyRadioButtonText	Only applies if the QlmEvaluationVisible property is set to true. Set the text in the radio button when the user has a license key.
QlmEvaluationLicenseKey	Only applies if the QlmEvaluationVisible property is set to true. Set the evaluation key to use when the user selects to evaluate the software and does not have a license key.
QlmEvaluationTrialChecked	Only applies if the QlmEvaluationVisible property is set to true. Checks the evaluation option by default.
QlmEvaluationTrialHelpText	Only applies if the QlmEvaluationVisible property is set to true. Set the text to display under the evaluation radio button.
QlmEvaluationTrialRadioButtonText	Only applies if the QlmEvaluationVisible property is set to true. Set the text to display on the evaluation radio button.
QlmEvaluationVisible	Enables the evaluation option. The evaluation option displays two radio buttons. One radio button allows the user to enter a license key and activate the license while the other radio button allows the user to evaluate the software by using an embedded evaluation license key.
QlmFormBackColor	Set the starting Background Color of the form to produce a gradient effect.
QlmFormBackColor2	Set the ending Background Color of the form to produce

	a gradient effect.
QlmGUID	Set the GUID associated to your product. The GUID can be found on the Define Product page in the QLM Console.
QlmHeaderBackColor	Set the Background Color of the header pane.
QlmLicenseStatus	Get the status of the license after it has been validated. This is a read-only property.
QlmLicenseType	Set the license type. The license type can be: ComputerName, UserDefined or Generic.
QlmLogoFont	Set the font to use in the logo text.
QlmLogoImage	Set the image to use for the logo.
QlmLogoText	Set the text to use for the logo.
QlmMajorVersion	Set the Major Version associated to your product. The Major Version can be found on the Define Products page in the QLM Console.
QlmMinorVersion	Set the Minor Version associated to your product. The Minor Version can be found on the Define Products page in the QLM Console.
QlmProductID	Set the Product ID Version associated to your product. The Product ID can be found on the Define Products page in the QLM Console.
QlmProductName	Set the Product Name associated to your product.
QlmPublicKey	Set the Public Key associated to your product. The Public Key Version can be found on the Define Products page (Keys tab) in the QLM Console.
QlmStoreKeysLocation	By default, QLM stores the license keys in a hidden file on the end user system. You can also select to store the license keys in the registry by setting this property.
QlmValidateCertificate	The QLM DLLs are digitally signed by a trusted certificate authority. In order to ensure that hackers do not replace the QLM DLLs by dummy ones, QLM can validate that the DLLs are properly signed.

Distribute your application

The sections below describe the options to include the QLM required DLLs in your application.

Select the most appropriate option based on the type of setup that you use to deploy your application.

For example, if you use a tool such as InstallShield to install your application, you should use the provided Merge Modules.

Special C

Using the QLM .NET API from VB6 or unmanaged C++

If you have included a reference to QlmLicenseLib.dll in your VB6, unmanaged C++ or any other non .NET based application, your setup must perform the following operations:

32 bit applications

- **Generate a type library to be refrenced by your code**
  %windir%\Microsoft.NET\Framework\v2.0.50727\regasm.exe /tlb "<fullpath>\QlmLicenseLib.dll"
- **Register the QlmLicenseLib.dll as a COM object**
  %windir%\Microsoft.NET\Framework\v2.0.50727\regasm.exe /codebase "<fullpath>\QlmLicenseLib.dll"

64 bit applications

- **Generate a type library to be refrenced by your code**
  "%windir%\Microsoft.NET\Framework64\v2.0.50727\regasm.exe" /tlb "<fullpath>\QlmLicenseLib.dll"
- **Register the QlmLicenseLib.dll as a COM object**
  "%windir%\Microsoft.NET\Framework64\v2.0.50727\regasm.exe" /codebase "<fullpath>\QlmLicenseLib.dll"

Distribute your application using Merge Modules

InstallShield and other MSI based installation tools

If you are using Windows Installer to distribute your application, you can use the merge modules found in the Quick License Manager **redistrib\MergeModules** folder. When including the merge module, set the destination folder of the merge module to be your application's installation folder.

Application Type	QLM Express - Merge Modules to include	QLM Professional or Enterprise - Merge Modules to include
Windows Forms .NET 2.0 or higher	Soraco Quick License Manager 5.0 (IsLicense50.msm) or (recommended) Soraco Quick License Manager 5.0 (IsLicense50.msm) Soraco Quick License Manager 5.0 for .NET 2.0 (QlmLicense.net2.msm) Soraco Quick License Manager User Controls 5.0 for .NET 2.0 (QlmControls.net2.msm)	Soraco Quick License Manager 5.0 (IsLicense50.msm) Soraco Quick License Manager 5.0 for .NET 2.0 (QlmLicense.net2.msm)
ASP.NET 2.0 or higher Excel 2003 or higher MS-Access 2003 or higher Outlook Add-in VB6	Soraco Quick License Manager 5.0 (IsLicense50.com.msm) or (recommended) Soraco Quick License Manager 5.0 for .NET 2.0 (QlmLicense.net2.emb.msm)/p>	Soraco Quick License Manager 5.0 for .NET 2.0 (QlmLicense.net2.emb.msm)
C++	Soraco Quick License Manager 5.0 (IsLicense50.com.msm) & or (recommended) Soraco Quick License Manager 5.0 (IsLicense50.msm)/p> Soraco Quick License Manager 5.0 for .NET 2.0 (QlmLicense.net2.com.msm)	Soraco Quick License Manager 5.0 (IsLicense50.msm) Soraco Quick License Manager 5.0 for .NET 2.0 (QlmLicense.net2.com.msm)

Note that if your application does not target x64 bit platforms, i.e. your application runs as a 32 bit application on 64 bit machines, then you should include the IsLicense50_x86.msm instead of IsLicense50.msm.

&

The IsLicense50.msm merge module includes 2 files:

- FileName: em>IsLicense50.dll
- Destination Folder: *[INSTALLDIR]*
- ComponentCode: *{CD707E5F-495E-48E5-8360-EAB26AFC186A}*
- Shared: *No*
- COM Extract at Build or Self-Register: *No*
- Permanent: *No*
- Architecture: *x86*
- FileName: *IsLicense50.dll*
- Destination Folder: *[INSTALLDIR]*
- ComponentCode: *{752E6A64-1248-46B5-87E5-8039A54F6288}*
- Shared: *No*
- COM Extract at Build or Self-Register: *No*
- Permanent: *No*
- Architecture: *x64*

The QlmLicenseLib.net2.msm merge module include several files:

- FileName: *QlmLicenseLib.dll*
- Destination Folder:*[INSTALLDIR]*
- ComponentCode: *{84A850A3-7CDE-4E23-B31E-58C4C716E54F}*

- Shared: *No*
- COM Extract at Build or Self-Register: *No*
- Permanent: *No*
- FileName: *QlmLicenseLib.resources.dll for Spanish*
- Destination Folder: *[INSTALLDIR]es*
- ComponentCode: *{560F660B-2649-45A9-BB53-D499DE808F34}*
- Shared: *No*
- COM Extract at Build or Self-Register: *No*
- Permanent: *No*
- FileName: *QlmLicenseLib.resources.dll for French*
- Destination Folder: *[INSTALLDIR]fr*
- ComponentCode: *{DC0FB90A-AE8A-4261-A82E-BC18B7DFB4C7}*
- Shared: *No*
- COM Extract at Build or Self-Register: *No*
- Permanent: *No*
- FileName: *QlmLicenseLib.resources.dll for Italian*
- Destination Folder: *[INSTALLDIR]it*
- ComponentCode: *{F040D81F-AC78-4CC9-9BD7-18B4F881F34F}*
- Shared: *No*
- COM Extract at Build or Self-Register: *No*
- Permanent: *No*&
- FileName: *QlmLicenseLib.resources.dll for German*
- Destination Folder: *[INSTALLDIR]de*
- ComponentCode: {7C02CB03-FB75-4B80-95C9-0F6421E578C1}
- Shared: *No*
- COM Extract at Build or Self-Register: *No*
- Permanent: *No*

Distribute your application using a Visual Studio Deployment Project

Visual Studio .Net Deployment Project

To add Quick License Manager to your Visual Studio .Net deployment project, follow the steps below:

- Create a deployment project
- Add your project's output in the File System Editor
- This should detect the Quick License Manager dependencies and include the following files:
  - IsLicense50.dll - If you are targeting x64 bit systems, you will need to conditionally install the proper IsLicense50.dll depending on the target platform.
    The x64 bit version of IsLicense50.dll is located in the Redistrib\x64 folder.
  - QlmLicenceLib.dll
  - QlmControls.dll
  - *es\QlmLicenseLib.resources.dll*
  - *fr\QlmLicenseLib.resources.dll*
  - *it\QlmLicenseLib.resources.dll*
  - *de\QlmLicenseLib.resources.dll*

Application Type	QLM Express - Files to include	QLM Professional or Enterprise - Files to include
Windows Forms .NET 2.0 or higher	redistrib\x86\IsLicense50.dll redistrib\x64\IsLicense50.dll or (recommended) redistrib\x86\IsLicense50.dll redistrib\x64\IsLicense50.dll redistrib\.net2.0\QlmLicenseLib.dll redistrib\.net2.0\QlmControls.dll	 redistrib\x86\IsLicense50.dll redistrib\x64\IsLicense50.dll redistrib\.net2.0\QlmLicenseLib.dll redistrib\.net2.0\QlmControls.dll
ASP.NET 2.0 or higher Excel 2003 or higher MS-Access 2003 or higher Outlook Add-in VB6	redistrib\x86\IsLicense50.dll redistrib\x64\IsLicense50.dll or (recommended) redistrib\.net2.0\QlmLicenseLibEmb\QlmLicenseLib.dll redistrib\.net2.0\QlmControls.dll	 redistrib\.net2.0\QlmLicenseLibEmb\QlmLicenseLib.dll redistrib\.net2.0\QlmControls.dll
C++	redistrib\x86\IsLicense50.dll redistrib\x64\IsLicense50.dll or (recommended) redistrib\x86\IsLicense50.dll redistrib\x64\IsLicense50.dll redistrib\.net2.0\QlmLicenseLib.dll redistrib\.net2.0\QlmControls.dll	 redistrib\x86\IsLicense50.dll redistrib\x64\IsLicense50.dll redistrib\.net2.0\QlmLicenseLib.dll redistrib\.net2.0\QlmControls.dll

Distribute your application using a manual procedure

Manual steps to install Quick License Manager files

If you are not using Windows Installer to distribute your application, the following files must be included in your application:

IsLicense50.dll: not shared, to be installed in the same folder as your application

QlmLicenseLib.dll: not shared, to be installed in the same folder as your application

QlmControls.dll: not shared, to be installed in the same folder as your application

If you are targeting x64 bit systems, you will need to conditionally install the proper IsLicense50.dll depending on the target platform.
The x64 bit version of IsLicense50.dll is located in the Redistrib\x64 folder.

Application Type	QLM Express - Files to include	QLM Professional or Enterprise - Files to include
Windows Forms .NET 2.0 or higher	redistrib\x86\IsLicense50.dll redistrib\x64\IsLicense50.dll or (recommended) redistrib\x86\IsLicense50.dll redistrib\x64\IsLicense50.dll redistrib\.net2.0\QlmLicenseLib.dll redistrib\.net2.0\QlmControls.dll	 redistrib\x86\IsLicense50.dll redistrib\x64\IsLicense50.dll redistrib\.net2.0\QlmLicenseLib.dll redistrib\.net2.0\QlmControls.dll
ASP.NET 2.0 or higher Excel 2003 or higher MS-Access 2003 or higher Outlook Add-in VB6	redistrib\x86\IsLicense50.dll redistrib\x64\IsLicense50.dll or (recommended) redistrib\.net2.0\QlmLicenseLibEmb\QlmLicenseLib.dll redistrib\.net2.0\QlmControls.dll	 redistrib\.net2.0\QlmLicenseLibEmb\QlmLicenseLib.dll redistrib\.net2.0\QlmControls.dll
C++	redistrib\x86\IsLicense50.dll redistrib\x64\IsLicense50.dll or (recommended) redistrib\x86\IsLicense50.dll redistrib\x64\IsLicense50.dll redistrib\.net2.0\QlmLicenseLib.dll redistrib\.net2.0\QlmControls.dll	 redistrib\x86\IsLicense50.dll redistrib\x64\IsLicense50.dll redistrib\.net2.0\QlmLicenseLib.dll redistrib\.net2.0\QlmControls.dll

Localization

QLM .NET API calls may return messages relating the status of a license key validation. By default, all messages returned by the QLM .NET API are in English.

The QLM .NET API also supports French, Spanish, Italian and German messages. To configure your application to display the proper message depending on the language of your customer's system, you need to:

- Locate the following folders in the Quick License Manager folder: es, fr, it, de
- When you deploy your application, copy these folders to the same location as the QlmLicenseLib.dll
- If your customer's system is running Spanish OS, the appropriate resources will be automatically loaded.

Alternatively, if you would like to force a specific language, you need to add the following call to your application prior to initializing any UI:

System.Threading.Thread.CurrentThread.CurrentUICulture = new System.Globalization.CultureInfo("es-ES");

64 bit Support

When distributing your application, depending on the QLM features that you are using, you may need to distribute the following DLLs with your application:

- IsLicense50.dll
- QlmLicenseLib.dll
- QlmControls.dll

IsLicense50.dll is a C++ DLL. If you are targeting a 64 bit platform, you need to ship with your application the 64 bit version of this DLL. The 64 bit version of the DLL is found in the x64 folder.

The *QlmLicenseLib.dll and QlmControls.dll* are .NET assemblies. The same version of these DLLs will work on 32bit or 64 bit platforms.

If your application is intended to run as 32 bit application on a x64 bit operating system, then you need to use the 32 bit version of IsLicense50.dll. If you are using the provided merge modules to integrate QLM into your setup, you should use the IsLicense50_x86.msm or IsLicense_x86.com.msm. These merge modules install the 32 bit version of the IsLicense50.dll regardless of the target platform.

BackwardCompatible

Set this property to true to allow validation of keys prior to the latest version of the QLM engine.

C++: VARIANT_BOOL BackwardCompatible

C#: bool BackwardCompatible

CreateLicenseKey

Creates a non-computer bound license key. If the ExpiryDate is NULL and the ExpiryDuration is -1, the license key is a permanent non-evaluation license key.

Prior to calling this function, you must call DefineProduct and set the PrivateKey property. Note that including the PrivateKey in your code is not recommended. Creation of license keys should not typically be done from within the application but rather from a server that the user does not have access to.

C++: _bstr_t CreateLicenseKey (DATE ExpiryDate, int ExpiryDuration)

C#: string CreateLicenseKey (System.DateTime ExpiryDate, int ExpiryDuration)

Parameters

ExpiryDate - The date when the license will expire. Use NULL if you do not want to specify an expiry date.

ExpiryDuration - The duration of the evaluation period in days. Use -1 if you do not want to specify a duration.

Return

A non-computer bound license key.

CreateLicenseKeyEx

Creates a computer bound license key.

Prior to calling this function, you must call DefineProduct and set the PrivateKey property. Note that including the PrivateKey in your code is not recommended. Creation of license keys should not typically be done from within the application but rather from a server that the user does not have access to.

C++: _bstr_t CreateLicenseKeyEx (ELicenseType LicenseType, BSTR MachineID)

C#: string CreateLicenseKeyEx (ELicenseType LicenseType, string MachineID)

Parameters

LicenseType - Specify the type of license to generate. See the definition of LicenseType below.

MachineID - A unique identifier for the machine. If you specify a *ComputerName* as the LicenseType, this argument must be the Computer Name. If you specify *User Defined* as the LicenseType, this argument can be anything you want. When validating the license key in your code, you will need to provide the same value to the ValidateLicenseEx function.

Return

A computer bound license key.

CreateLicenseKeyEx2

Creates a computer bound license key that has an expiry date and a number of licenses.

Prior to calling this function, you must call DefineProduct and set the PrivateKey property. Note that including the PrivateKey in your code is not recommended. Creation of license keys should not typically be done from within the application but rather from a server that the user does not have access to.

C++: _bstr_t CreateLicenseKeyEx2 (DATE ExpiryDate, int ExpiryDuration, int NumberOfLicenses, ELicenseType LicenseType, BSTR MachineID)

C#: string CreateLicenseKeyEx2 (System.DateTime ExpiryDate, int ExpiryDuration, int NumberOfLicenses, ELicenseType LicenseType, string MachineID)

Parameters

ExpiryDate - The date when the license will expire. Use NULL if you do not want to specify an expiry date.

ExpiryDuration - The duration of the evaluation period in days. Use -1 if you do not want to specify a duration.

NumberOfLicenses - The number of licenses associated with the key. Use 1 if you do not want to use single activation licensing.

LicenseType - Specify the type of license to generate. See the definition of LicenseType below.

MachineID - A unique identifier for the machine. If you specify a *ComputerName* as the LicenseType, this argument must be the Computer Name. If you specify *User Defined* as the LicenseType, this argument can be anything you want. When validating the license key in your code, you will need to provide the same value to the ValidateLicenseEx function. To create a key that is not computer bound, set this argument to NULL and set the LicenseType to Generic.

Return

A computer bound license key.

CreateLicenseKeyEx3

Creates a computer bound license key that has an expiry date, a number of licenses and a specific set of features that are enabled.

Prior to calling this function, you must call DefineProduct and set the PrivateKey property. Note that including the PrivateKey in your code is not recommended. Creation of license keys should not typically be done from within the application but rather from a server that the user does not have access to.

C++: _bstr_t CreateLicenseKeyEx3 (DATE expiryDate, int expiryDuration, int numberOfLicenses, ELicenseType licenseType, BSTR machineID, in features)

C#: string CreateLicenseKeyEx3 (System.DateTime expiryDate, int expiryDuration, int numberOfLicenses, ELicenseType licenseType, string machineID, int features)

Parameters

ExpiryDate - The date when the license will expire. Use NULL if you do not want to specify an expiry date.

ExpiryDuration - The duration of the evaluation period in days. Use -1 if you do not want to specify a duration.

NumberOfLicenses - The number of licenses associated with the key. Use 1 if you do not want to use single activation licensing.

LicenseType - Specify the type of license to generate. See the definition of LicenseType below.

MachineID - A unique identifier for the machine. If you specify a *ComputerName* as the LicenseType, this argument must be the Computer Name. If you specify *User Defined* as the LicenseType, this argument can be anything you want. When validating the license key in your code, you will need to provide the same value to the ValidateLicenseEx function. To create a key that is not computer bound, set this argument to NULL and set the LicenseType to Generic.

Features - A value specifying the features that should be enabled in the created key. Each feature has a unique ID associated to it. To combine features, perform a bitwise OR operation on the required features.

Return

A computer bound license key.

CreateLicenseKeyEx4

Creates a computer bound license key that has an expiry date, a number of licenses and a specific set of features that are enabled.

Prior to calling this function, you must call DefineProduct and set the PrivateKey property. Note that including the PrivateKey in your code is not recommended. Creation of license keys should not typically be done from within the application but rather from a server that the user does not have access to.

C++: _bstr_t CreateLicenseKeyEx4 (DATE expiryDate, int expiryDuration, int numberOfLicenses, ELicenseType licenseType, BSTR machineID, int[] features)

C#: string CreateLicenseKeyEx4 (System.DateTime expiryDate, int expiryDuration, int numberOfLicenses, ELicenseType licenseType, string machineID, SAFEARRAY *Features)

Parameters

ExpiryDate - The date when the license will expire. Use NULL if you do not want to specify an expiry date.

ExpiryDuration - The duration of the evaluation period in days. Use -1 if you do not want to specify a duration.

NumberOfLicenses - The number of licenses associated with the key. Use 1 if you do not want to use single activation licensing.

LicenseType - Specify the type of license to generate. See the definition of LicenseType below.

MachineID - A unique identifier for the machine. If you specify a *ComputerName* as the LicenseType, this argument must be the Computer Name. If you specify *User Defined* as the LicenseType, this argument can be anything you want. When validating the license key in your code, you will need to provide the same value to the ValidateLicenseEx function. To create a key that is not computer bound, set this argument to NULL and set the LicenseType to Generic.

Features - An array of feature sets. Each feature set is a value specifying the features that should be enabled in the created key. The value of the feature set is the or'ed value of all the features to be enabled in the set. To combine features, perform a bitwise OR operation on the required features.

Return

A computer bound license key.

CreateLicenseKeyEx5

Creates a computer bound license key that has an expiry date, a number of licenses and a specific set of features that are enabled. This API is functionally identical to CreateLicenseKeyEx4. It was created to accomodate programming languages such as VB6 that cannot handle the array data type used in CreateLicenseKeyEx4.

Prior to calling this function, you must call DefineProduct and set the PrivateKey property. Note that including the PrivateKey in your code is not recommended. Creation of license keys should not typically be done from within the application but rather from a server that the user does not have access to.

C++: _bstr_t CreateLicenseKeyEx5 (DATE expiryDate, int expiryDuration, int numberOfLicenses, ELicenseType licenseType, BSTR machineID, BSTR features)

C#: string CreateLicenseKeyEx5 (System.DateTime expiryDate, int expiryDuration, int numberOfLicenses, ELicenseType licenseType, string machineID, string Features)

Parameters

ExpiryDate - The date when the license will expire. Use NULL if you do not want to specify an expiry date.

ExpiryDuration - The duration of the evaluation period in days. Use -1 if you do not want to specify a duration.

NumberOfLicenses - The number of licenses associated with the key. Use 1 if you do not want to use single activation licensing.

LicenseType - Specify the type of license to generate. See the definition of LicenseType below.

MachineID - A unique identifier for the machine. If you specify a *ComputerName* as the LicenseType, this argument must be the Computer Name. If you specify *User Defined* as the LicenseType, this argument can be anything you want. When validating the license key in your code, you will need to provide the same value to the ValidateLicenseEx function. To create a key that is not computer bound, set this argument to NULL and set the LicenseType to Generic.

Features - A set of features to be enabled using the following syntax:

<featureSet>:<featureValue>;<featureSet>:<featureValue>
Example: "0:8;1:2;3:14" - Enables: feature 8 in feature set 0, feature 2 in feature set 1 and features 2, 4, 8 in feature set 3.
To combine features, perform a bitwise OR operation on the required features.

Return

A computer bound license key.

DaysLeft

Returns the number of days left before the evaluation expires. You must call ValidateLicense prior to calling this function.

C++: int DaysLeft

C#: int DaysLeft ()

Return

Number of days left before the evaluation expires.

DefineProduct

The DefineProduct method initializes basic information required to validate license keys. You must call this function prior to calling any other function.

C++: HRESULT DefineProduct (int ProductID, BSTR ProductName, int MajorVersion, int MinorVersion, BSTR EncryptionKey, BSTR PersistenceKey)

C#: DefineProduct (int ProductID, string ProductName, int MajorVersion, int MinorVersion, string EncryptionKey, string PersistenceKey)

Parameters

ProductID: ID of the product as generated by Quick License Manager

ProductName: Name of the product

MajorVersion: Major version of the product (maximum 2 digits)

MinorVersion: Minor version of the product (maximum 2 digits)

Encryption Key: string used to encrypt the license key.

PersistenceKey: GUID associated with the product and automatically generated by Quick License Manager for each product. The evaluation information of the product is stored at runtime in the registry under HKCR\CLSID\<GUID>.

Duration

Returns the duration in days of the evaluation key. You must call ValidateLicense prior to calling this function.

C++: int Duration

C#: int Duration ()

Return

Duration of the evaluation key.

ELicenseStatus

Enum of all possible values of the license key status. Note that the status can consist of a combination of these values:

EKeyPermanent : The license key is valid and it is a permanent license key.

EKeyInvalid : The license key is invalid. It was not decoded succesfully.

EKeyDemo : The license key is an evaluation key.

EKeyProductInvalid : The product ID of the license key does not correspond to the expected Product ID.

EKeyVersionInvalid : The Major or Minor version of the license key does not correspond to the expected Major or Minor version.

EKeyExpired : The license key has expired.

EKeyTampered: The license key was tampered typically indicating that the user is attempting to set the date back to run the software.

EKeyMachineInvalid: If you are using computer bound license keys, an EKeyMachineInvalid status indicates that the license key that was validated does not match the computer to which the license key was bound.

EKeyNeedsActivation: This flag indicates that the license key is an activation key. If you detect an activation key, you should not enable your application. You should just allow the user to activate his license. Once the license is activated, a computer bound key is issued. Once you detect a valid computer bound key, you can enable your application.

ELicenseType

Enum of all possible types of license keys.

Activation : The license key is a key that requires activation.

Evaluation (obsolete) : The license key is an evaluation key.

ComputerName : The license key is bound to the name of the computer.

Generic (previously PermanentGeneric) : The license key is permanent and not bound to a computer.

UserDefined: The license key is bound to the computer based on a user defined unique identifier.

EvaluationPerUser

Set this property to true to store evaluation information per user. The default value is true. If set to false, evaluation information is stored at the machine level. Note that you need to make sure the current user has the required privileges to store evaluation information at the machine level under HKEY_LOCAL_MACHINE\Software\Classes\CLSID\<GUID>.

Evaluation information consists of the installation date of your software as well as the last time your software ran.

C++: VARIANT_BOOL EvaluationPerUser

C#: bool EvaluationPerUser

ExpiryDate

Returns the expiry date of the evaluation key. You must call ValidateLicense prior to calling this function.

C++: DATE ExpiryDate

C#: System.DateTime ExpiryDate()

Return

Expiry date of the evaluation key.

Features

Returns an array of all the feature sets associated with the license key. Within a feature set (each element of the array), if several features are associated to a license key, the returned value is a bitwise OR of these features.

This function must be called after a call to ValidateLicense or ValidateLicenseEx.

C++: int *Features

C#:/STRONG> int [] Features

GetStatus

Returns the last status. See ELicenseStatus for possible values.

You must always call this function after calling ValidateLicense or ValidateLicenseEx to get the result of the validation.

C++: int GetStatus

C#: int GetStatus ()

Return

Last status

IsEvaluation

Returns whether the current license key is an evaluation key. You must call ValidateLicense prior to calling IsEvaluation.

C++: VARIANT_BOOL IsEvaluation

C#: bool IsEvaluation ()

Return

Boolean indicating if the license key is an evaluation key.

IsFeatureEnabled

Returns whether the specified feature is enabled in this license key. This function is now obsolete and has been superseded by IsFeatureEnabledEx.

You must call ValidateLicense prior to calling IsFeatureEnabled. **C++:** VARIANT_BOOL IsFeatureEnabled (int feature)

C#: bool IsFeatureEnabled (int feature)

Parameters

feature - id of feature to be checked.

Return

Boolean indicating if the featured is enabled.

IsFeatureEnabledEx

Returns whether the specified feature is enabled in this license key. You must call ValidateLicense prior to calling IsFeatureEnabled.

C++: VARIANT_BOOL IsFeatureEnabled (int featureSet, int feature)

C#: bool IsFeatureEnabledEx (int featureSet, int feature)

Parameters

featureSet - id of the feature set. QLM supports four feature sets (0 to 3).

feature - id of feature to be checked.

Return

Boolean indicating if the featured is enabled.

IsValid

Returns whether the current license key is a valid key. A valid license key is a key that was decoded properly and is either permanent or evaluation. You must call ValidateLicense prior to calling IsValid.

C++: VARIANT_BOOL IsValid

C#: bool IsValid ()

Return

Boolean indicating if the license key is a valid key.

LicenseType

Returns the license type of the key. See ELicenseType for possible values.

C++: ELicenseType LicenseType

C#: ELicenseType LicenseType ()

Return

License type

NumberOfLicenses

Returns the number of multiple activations enabled for the license key.

C++: int NumberOfLicenses

C#: int NumberOfLicenses;

Return

Number Of LNumber Of Licenses

MajorVersion

Returns the major version associated to the license key. You must call ValidateLicense prior to calling this function.

C++: int MajorVersion

C#: int MajorVersion

Return

Major version of the product.

MinorVersion

Returns the minor version associated to the license key. You must call ValidateLicense prior to calling this function.

C++: int MinorVersion

C#: int MinorVersion

Return

Minor version of the product.

PrivateKey

QLM version 5 implements asymmetric encryption to encrypt the license key. Asymmetric enryption is safer because one key encrypts the license, the private key, and another key, the public key, decrypts that information. Therefore, you only need to include the public key in your source code.

This function sets the private key associated with your product. The private key should be set before you create a license, typically right after the call to DefineProduct. If you are creating a license key with a QLM engine version prior to version 5, you do not need to set the private key. It is highly recommended that you do not set the private key in your code.

The private key of your product can be found on the DefineProduct screen under the Keys tab in the QLM Console.

C++: _bstr_t privateKey

C#: string PrivateKey

PublicKey

QLM version 5 implements asymmetric encryption to encrypt the license key. Asymmetric enryption is safer because one key encrypts the license, the private key, and another key, the public key, decrypts that information. Therefore, you only need to include the public key in your source code.

This function sets the public key associated with your product. The public key should be set before you validate a license, typically right after the call to DefineProduct. If you are validating a license key with a QLM engine version prior to version 5, you do not need to set the public key.

The public key of your product can be found on the DefineProduct screen under the Keys tab in the QLM Console.

C++: _bstr_t publicKey

C#: string PublicKey

ProductID

Returns the product ID associated to the license key. You must call ValidateLicense prior to calling this function.

C++: ibt ProductID

C#: int ProductID ()

Return

Product ID associated to the license key.

ValidateLicense

Validates a license key. You must call DefineProduct prior to calling this function.

C++: _bstr_t ValidateLicense (BSTR LicenseKey);

C#: string ValidateLicense (string LicenseKey)

Parameters

LicenseKey: License Key to validate

Return

Error message if ValidateLicense fails to validate or if the license is an evaluation license.

ValidateLicenseEx

Validates a computer bound license key. You can call this function for any type of license key. If the license key is not computer bound, set the ComputerID to an empty string. You must call DefineProduct prior to calling this function.

C++: _bstr_t ValidateLicenseEx (BSTR LicenseKey, BSTR ComputerID);

C#: string ValidateLicenseEx (string LicenseKey, string ComputerID)

Parameters

LicenseKey: License Key to validate

ComputerID: A string identifying the computer. License Key to validate

Return

Error message if ValidateLicenseEx fails to validate or if the license is an evaluation license.

ValidateFile

Validates that the Quick License Manager DLL is authentic and was not tampered with. In order to prevent hackers from replacing the IsLicense50.dll with their own version, you can validate the authenticity of the DLL by calling the ValidateFile function. The ValidateFile function returns a fingerprint (long number) that is the result of a checksum of the DLL contents combined with your own key. Use the QlmFingerPrint.exe to generate this unique fingerprint and validate in your code that the runtime fingerprint matches the generated one.

If you are using QLM Professional, you do not need to call this function. Instead, set the validateIntegrity argument to true when constructing the QlmLicense object.

C++: long ValidateFile (BSTR LicenseDLL, BSTR Key);

C#: ulong ValidateFile(string LicenseDLL, BSTR Key)

Parameters

LicenseDLL: Full path to the License DLL. If this argument is NULL, the currently loaded License DLL is used.

Key: A unique key of your choice that is used to uniquely encrypt the fingerprint.

Return

FingerPrint- A long umber that uniquely identifies your license DLL.

Version

Returns the version of the QLM engine used to create the key. You must call ValidateLicense prior to calling this function.

C++: _bstr_t Version

C#: string Version

Return

Version of QLM Engine used to create the license key.

Quick License Manager Professional Overview

Quick License Manager Professional provides the tools required to implement online software activation. QLM Professional is composed of 2 components: (a) a web service that provides interfaces for issuing and managing license keys and (b) a client application that communicates with the web service and provides a user interface to the web service.

Software Activation is the process of generating a computer bound license key over the internet. In order to implement software activation with Quick License Manager, you need to work with 2 types of license keys.

The first key is called "Activation Key" (ELicenseType.Activation). This is the key that you send to your customer when they purchase your software. The activation key can be sent to your customer directly from your eCommerce provider (see other section on how to integrate QLM with your eCommerce provider) or manually. This key does not enable your software. It simply allows the user to activate his license. If a user purchases several copies of your software, you can send them one activation key for all purchased licenses or one activation key per license.

From your application, the user enters the activation key. Your application then calls the QLM API which sends the activation key along with a with a computer unique identifier to the QLM Web service. The QLM Web service then generates the second type of license key called "Computer Bound" (ELicenseType.UserDefined or ELicenseType.ComputerName) and sends this key back to your application. The computer bound key enables your application.

Note that in your application, you should create a dialog for the user to enter a license key and activate it. Alternatively, you can use the QLM User Control (.NET) which provides a ready-made control for entering and processing the activation key.

Web Service

The QLM web service stores all license keys in a database. The default database that ships with QLM is a MS-Access database. However, you can use any database that implements an OleDbProvider. See the Configuring your database section below for more details. The officially supported databases are MS-SQL Server 2000 and higher and MS-Access.

The system requirements for the QLM web service are:

- Windows 2003 server or higher (x86 or x64).
- .NET Framework 3.5
- SQL Server Database or MS-Access
- Full Trust for .NET assemblies.

The web service can be installed in 2 ways: (a) by running the provided setup program **qlmwebsvcsetup.exe** or (b) by executing the installation steps manually. If you are hosting your own web site or if you are distributing the QLM Web service as part of your program (requires special distribution rights), then option (a) is recommended. If your site is hosted at an ISP, then you may need to use option (b).

Automated installation of the web service

To install the QLM Web service, locate the **qlmwebsvcsetup.exe** setup program in the QLM installation folder. Typically this file is located under:

C:\Program Files\Soraco\QuickLicenseMgr\DeployToServer

Execute the setup and follow the onscreen instructions.

Manual installation of the web service

To manually install the QLM Web service, locate the **QlmWebService** folder in the QLM installation folder. Typically this folder is located under:

C:\Program Files\Soraco\QuickLicenseMgr\DeployToServer\QlmWebService

- At your ISP, create a new virtual directory called **qlm** and enable ASP.NET 2.0 for this virtual directory. Configure this virtual directory as an Application.
- Upload all the files in the DeployToServer\QlmWebService folder and subfolders to the virtual directory (preserve the directory structure).

Configure the web service

- Customize the following settings in the web.config files based on your needs:
  - Database connection string (refer to the Configure the Database section in the Help)
  - Default QLM Engine Version (defaultQlmVersion).
  - SQL Syntax(sqlsyntax).
  - Communication Encryption Key (communicationEncryptionKey). The communication encryption key is used to encrypt data transferred between QLM and the QLM Web service. This key is like a password that protects your data. You should use a value that is hard to guess and at least 8 character long.
  - Admin Encryption Key (adminEncryptionKey). The admin encryption key is used to encrypt data transferred between QLM and the QLM Web service. This key is like a password that protects your data. You should use a value that is hard to guess and at least 8 character long.

Security Note:

- You need to give the anonymous user (IUSR_XXX, IWAM_XXX) execute privileges to the bin folder.

Recommendations

- Change the default Communication and Admin Encryption Keys. If you do not, any other QLM customer may be able to view your data.

Configuring the Database

QLM Professional stores all issued license keys as well as customer related information in a database on the web server. The default database that ships with QLM is a MS-Access database. However, you can use any database that implements an OleDbProvider.

Database Installation

If you installed the QLM web service using the provided setup program, the database is created automatically during the setup.

However, if you manually installed the QLM web service, you will need to create the database manually as well as follows:

- If you want to use the MS-Access Database, copy the qlm.mdb file from C:\Program Files\Soraco\QuickLicenseMgr\DeployToServer\QlmWebService\Db to the location recommended by your ISP and update the web.config file accordingly. Note that the web.config must contain the appropriate local path to the qlm.mdb file.
- If you want to use an SQL Server database, use the tools provided by your ISP to create a database called Qlm (or any other name of your choice) or execute the sql200x.createdb.sql.
- Execute the provided database creation script sql200x.createtables.sql.
- Execute the provided database creation script sql200x.createusers.sql.
- Update the web.config file to point to the SQL database (see comments in web.config and section below).

Configure QLM to use a SQL Server Database

To use a database engine other than MS-Access, you need to update the connectionString and the sql Syntax settings in the web.config file that is included with the web service as follows:

For all SQL Server editions, locate the sqlSyntax setting in the web.config and set it as follows:

```
<setting name="sqlSyntax" serializeAs="String">
<value>sql</value>
</settings>
```

For the connectionStrings, locate in the web.config a commented connectionString section that corresponds to the type of database you are using. Uncomment the section and update the connectionString settings accordingly.

Recommendations

- Backup your database on a daily basis.

Multiple Activations Keys

When customers purchase several copies of your product, there are two ways you can send them their license keys.

1. You can send them one license key for each purchased copy. For example, if a customer purchases 5 copies of your software, you will send them 5 license keys. This approach is inconvenient both for the seller who has to manage multiple keys per customer and the buyer who has to associate a key per computer.

2. You can send them one license key that can be activated on 5 computers. This approach is simple both for the vendor and the seller. This method is referred to in QLM as **Multiple Activations Key**.

How To create a license key using Multiple Activations

From the QLM Console, click on Manage Keys / License Keys / Create. In the Create License dialog, select the Multiple Activations key check box and specify the number of activations.

If you are using an http request such as GetActivationKey to create activation keys, add the is_usemultipleactivationskey=true argument to the URL. When integrated with an eCommerce provider, the number of activations is typically determined based on the number of copies purchased. Alternatively, you can add the is_quantity argument in the URL request.

When a license key using Multiple Activations is activated, QLM maintains the activation information in a separate table called ActivationLog.

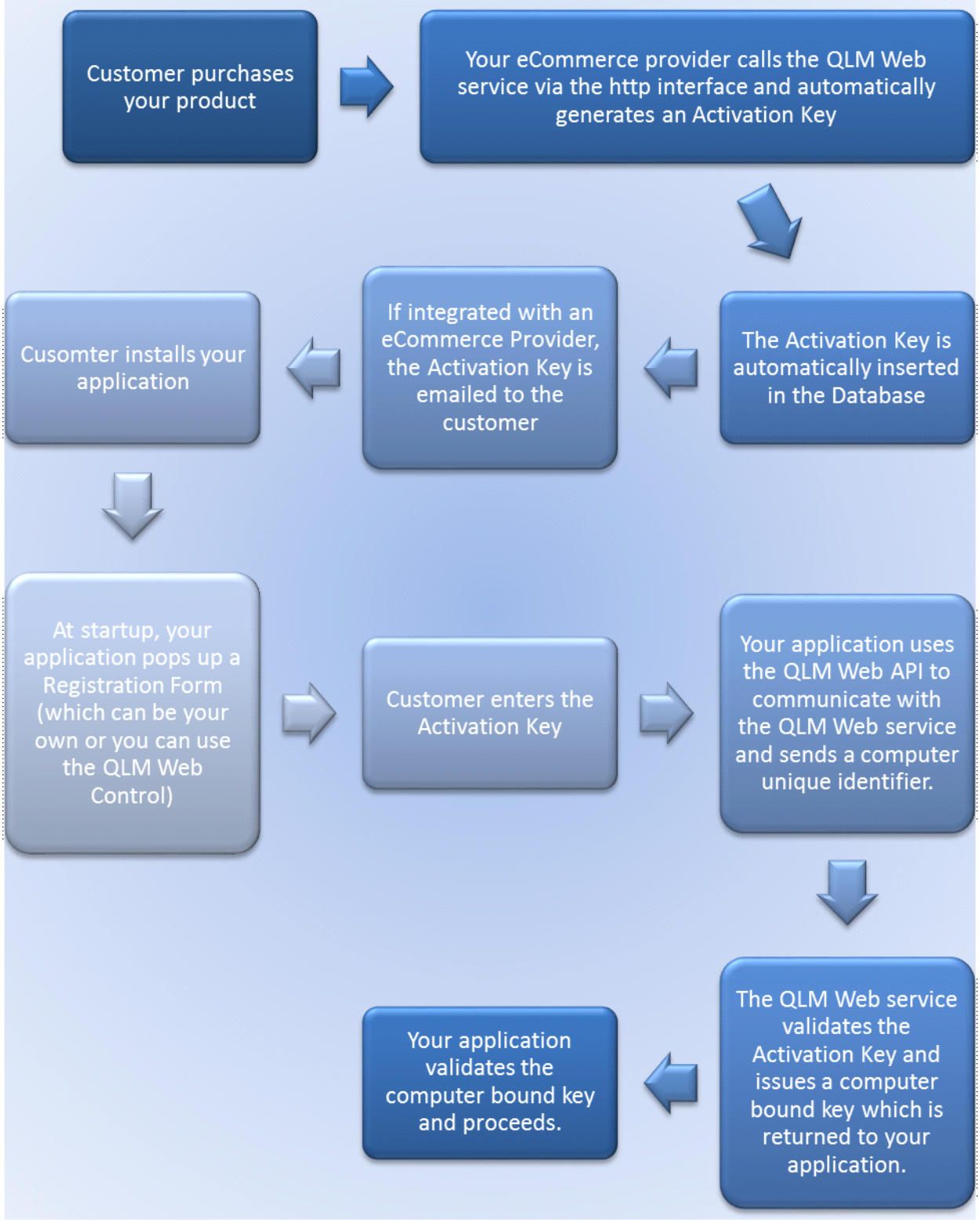
To view the activated licenses, start QLM, click on Manage Keys, run a search to display a set of records. Licenses that are of type Multiple Activations display a + sign in the License Type field. Click on the + sign to expand the row and view the activated licenses.

You can perform one of three operations on the multiple activations licenses:

Release	Releases an activated license.
Edit	Edits an activated license.
Delete	Deletes an activated license.

QUICK LICENSE MANAGER WEB ONLINE ACTIVATION

[Use this method if your eCommerce provider can call a web service]



QUICK LICENSE MANAGER WEB ONLINE ACTIVATION

[Use this method if your eCommerce provider cannot call a web service]

Online Activation using the QLM .NET Basic Activation Control

QLM provides a control that can be dropped in your application to simplify the process of online activation. When using the control, there is almost no need to write any code to implement online activation.

The control is available for applications developed using Microsoft .NET 2.0 or later.

A sample program QlmWebControlSample1 can be found in the samples folder.

To use the Qlm Web Control:

- Create a form in your application.
- Right-mouse click on the Visual Studio Toolbox and select "Choose Items".
- Locate and select the QlmControls.dll in the Quick License Manager folder.
- A new toolbox item called QlmWebBasicActivationControl should appear.
- Drag this item and drop it onto your form.
- Add a reference to QlmLicenseLib.dll.
- Edit the properties of this control and locate all the QLM properties (prefixed with Qlm). In the Properties window, click on the Categorized button to view the QLM properties in categories.
- Click on each QLM property to view help about the property.
- Before testing, make sure to copy the IsLicense50.dll to your Debug or Release folder.

The QlmWebBasicActivationControl also exposes 2 events:

- QlmClose is triggered when the Close button is clicked
- QlmActivate is triggered when the Activate button is clicked

For more details about all the properties exposed by the .NET Control, review the .NET Control Reference section in the Help.

In addition to integrating the control, you need to modify your application to validate a license key at startup. Once the user has activated his license, the control stores the license key in a hidden location on the end user system. At startup, you should validate that the stored license key is still valid. Use the QLM Code Generator to generate the LicenseValidator class. This class has a method called ValidateLicenseAtStartup which you can call when your application is launched:

```
LicenseValidator lv = new LicenseValidator ();

if (lv.ValidateLicenseAtStartup(computerID, ref needsActivation, ref returnMsg) == false)

{

// the stored license key is not valid. Display the Qlm Control to allow the user to enter a new key

}
```

Step by Step Procedure to implement Software Activation

- Define a Product in Quick License Manager.
- Install the QLM web service
- Configure the QLM Console to connect to the QLM Web service
- Modify your application by including the Qlm Web control as described above
- Make sure to include the following DLLs in your application: QlmControls.dll, QlmLicenseLib.dll and IsLicense50.dll.
- Modify your application to validate a license key at startup.
- If you use a third-party e-commerce provider to process your orders, have that provider invoke the GetActivationKey method (see online activation using http request) when an order is placed so that the email the customer receives contains an Activation key.

Implement Online Activation using the QLM License Wizard .NET Control

QLM provides a control that can be dropped in your application to simplify the process of online activation. When using the control, there is almost no need to write any code to implement online activation.

The control is available for applications developed using Microsoft .NET 2.0 or later.

A sample program QlmLicenseWizardSample can be found in the samples folder.

To use the QLM License Wizard .NET Control:

- Create a form in your application.
- Locate the Quick License Manager section in your Visual Studio toolbox. If you cannot locate it, do the following:
  - Right-mouse click on the Visual Studio Toolbox and select "Choose Items".
  - Locate and select the QlmControlLicenseWizard.dll in the Quick License Manager folder.
  - A new toolbox item called QlmLicenseWizardCtrl should appear.
- Drag this item and drop it onto your form.
- Add a reference to QlmLicenseLib.dll.
- Edit the properties of this control and locate all the QLM properties (prefixed with Qlm). In the Properties window, click on the Categorized button to view the QLM properties in categories.
- Click on each QLM property to view help about the property.
- Before testing, make sure to copy the IsLicense50.dll to your Debug or Release folder.

The QlmLicenseWizardCtrl also exposes 3 events:

- QlmClose is triggered when the Close button is clicked
- QlmActivate is triggered when the Activate button is clicked
- QlmCancel is triggered when the Cancel button is clicked

For more details about all the properties exposed by the .NET Control, review the .NET Control Reference section in the Help.

In addition to integrating the control, you need to modify your application to validate a license key at startup. Once the user has activated his license, the control stores the license key in a hidden location on the end user system. At startup, you should validate that the stored license key is still valid. Use the QLM Code Generator to generate the LicenseValidator class. This class has a method called ValidateLicenseAtStartup which you can call when your application is launched:

```
LicenseValidator lv = new LicenseValidator ();

while (lv.ValidateLicenseAtStartup(computerID, ref needsActivation, ref returnMsg) == false)

{

// the stored license key is not valid. Display the Qlm License Wizard Control to allow the user to enter a new key

}
```

Step by Step Procedure to implement Software Activation

- Define a Product in Quick License Manager.
- Install the QLM web service.
- Configure the QLM Console to connect to the QLM Web service.
- Modify your application by including the QLM .NET control as described above.
- Make sure to include the following DLLs in your application: QlmControlLicenseWizard.dll, QlmLicenseLib.dll, x86\IsLicense50.dll and x64\IsLicense50.dll.
- Modify your application to validate a license key at startup.
- If you use a third-party e-commerce provider to process your orders, have that provider invoke the GetActivationKey method (see online activation using http request) when an order is placed so that the email the customer receives contains an Activation key.

Online Activation using the QLM License Wizard Standalone Application

QLM provides a standalone application (QlmLicenseWizard.exe) that you can invoke from your application to simplify the process of online activation. When using QlmLicenseWizard.exe, there is almost no need to write any code to implement online activation. This method is ideal for non .NET applications that cannot use the QLM .NET Control. Samples are provided for C++, MS-Access and Excel.

The QlmLicenseWizard.exe can be used on any Windows XP and higher system and requires Microsoft .NET 2.0 or later.

To use the QlmLicenseWizard.exe application:

- From the QLM Console, create a product.
- Go to the **Protect your application** tab and follow the instructions in the wizard.
- Modify your application as follows:
    - When your application is launched, call the *ValidateLicenseAtStartup* function. This function is generated by the **Protect your application** wizard in the *LicenseValidator* class.
    - If the ValidateLicenseAtStartup function returns false, launch the QlmLicenseWizard.exe with the /settings and /uiSettings arguments.
    - When the QlmLicenseWizard.exe process exits, call *ValidateLicenseAtStartup* again until the license is valid.

Once the user has activated his license from the QlmLicenseWizard process, the license key is automatically stored in a hidden location on the end user system.

Example:

```
LicenseValidator lv = new LicenseValidator ();

while (lv.ValidateLicenseAtStartup(computerID, ref needsActivation, ref returnMsg) == false)

{

    // Launch the QlmLicenseWizard.exe process

   // If the process exit code is 4, break the while loop

}
```

The command line arguments of QlmLicenseWizard.exe are:

Argument	Meaning
/settings	Path to the settings file generated by the **Protect your application** wizard (enclose the full path in double quotes).
/uiSettings	Path to the UI settings file generated by the **Protect your application**
/computerID	The unique identifier of the current system. If not argument is provided, the computer name is used.
/activationKey	Optional argument. By default, the activation key is automatically retrieved from the location associated with your products by calling the ReadKeys API. To override this behavior, you can specify the activationKey on the command line.
/computerKey	Optional argument. By default, the computer key is automatically retrieved from the location associated with your products by calling the ReadKeys API. To override this behavior, you can specify the activationKey on the command line.

Upon exiting, the QlmLicenseWizard.exe process returns the following exit codes:

Exit Code	Meaning
0	the license is valid

1	the license is not valid
2	the license has expired
3	usage error in the command line, typically indicating some missing or invalid arguments
4	the user cancelled the wizard

Step by Step Procedure to implement Software Activation

- Define a Product in Quick License Manager..
- Install the QLM web service.
- Configure the QLM Console to connect to the QLM Web service.
- Modify your application by invoking the QlmLicenseWizard.exe as described above.
- Make sure to include the following DLLs in your application: QlmLicenseLib.dll, x86\IsLicense50.dll and x64\IsLicense50.dll.

Online Activation using API calls

To activate a license key, you invoke the ActivateLicense or ActivateLicenseForUser methods. ActivateLicense should be called when the activation key is already associated to a user. ActivateLicenseForUser should be called to activate the license and associate it to the specified user.

Note that to add a user, you can call the AddUser API or call the ActivateLicenseDialog which display a license registration form that allows the user to register the license.

Example:

```
QlmLicense lic = new QlmLicense();
lic.CommunicationEncryptionKey = "{B6163D99-F46A-4580-BB42-BF276A507A14}";
lic.DefineProduct(1, "My Product", 1, 1, "encKey", "{549D9583-7152-41bf-9322-AF0A6DB28223}");
lic.ActivateLicense("http://localhost/qlmservice.asmx", activationKey, computerKey, computerName, "5.0.00",
"my data", out response);

ILicenseInfo licenseInfo = new LicenseInfo();
string message = string.Empty;
if (target.ParseResults(response, ref licenseInfo, out message))
{
Console.WriteLine ("PC key=" + licenseInfo.ComputerKey);
}
```

Step by Step Procedure to implement Software Activation

- Define a Product in Quick License Manager.
- Install the QLM web service
- Configure the QLM Console to connect to the QLM web service
- Modify your application so that you can handle the following scenarios:
  - If an evaluation key is entered, you process it without activation.
  - If an activation key is entered, you invoke your web service to get a machine bound key. Once you have received a computer bound key, store it along with the activation key with your application's settings. When your application is started, load these 2 keys. Validate the computer bound key using ValidateLicenseEx. If the key validates properly, enable your software. If it fails, display the activation key and allow the user to activate his license again.
  - If a machine bound key is entered, you validate that it matches the current computer.
  - If a Generic key is entered, you enable your software without activation. Note that even if you implement software activation, you may want to support Generic keys for Corporate customers that purchase site licenses. Software Activation in this situation may not be desirable.
- If you use a third-party e-commerce provider to process your orders, have that provider invoke the GetActivationKey method (see online activation using http request) when an order is placed so that the email the customer receives contains an Activation key.
- When the user enters the Activation key in your application, you invoke the ActivateKey method providing the computer identifier to the service. A computer bound license key is returned and the software is enabled. The ActivateKey method can be invoked in 2 ways: (a) by using the URL syntax or (b) by calling the ActivateLicense method that is available in the QlmLicenseLib assembly. For more details, see the **Activating a license key** section below.

If you need to create an activation key from your web site, use the CreateActivationKey method. This method will create an activation key and store it in the database. Note that prior to calling CreateActivationKey, you must call the DefineProduct function.

In addition, this function can only be called if you update the web.config on the web server as follows:

```
<setting name="enableCreateActivationKey" serializeAs="String">
<value<True</value>
</setting>
```

For security reasons, creating activation keys should always be done from your site and not integrated in your application.

Example:

```
QlmLicense lic = new QlmLicense();
lic.CommunicationEncryptionKey = "{B6163D99-F46A-4580-BB42-BF276A507A14}";
lic.DefineProduct(1, "My Product", 1, 1, "encKey", "{549D9583-7152-41bf-9322-AF0A6DB28223}");
lic.CreateActivationKey("http://localhost/qlmservice.asmx", "john@sm.com",
255, 1, true, "5.0.00", string.Empty, "my data",
out response);

ILicenseInfo licenseInfo = new LicenseInfo();
string message = string.Empty;
if (target.ParseResults(response, ref licenseInfo, out message))
```

```
{
Console.WriteLine ("Activation key=" + licenseInfo.ActivationKey);
}
```

Online Activation using http requests

To activate a license using an http request, use the following command:

http://yourserver/qlm/qlmservice.asmx/ActivateKey?is_productid=4&is_majorversion=3&is_minorversion=0&is_pcid=00-30-BD-92-74-25&is_avkey=A2CC8-0F116-8EA0A-CC2C10

where is_pcid is the unique identifier for a PC and is_avKey is the value returned from GetActivationKey.

For details about the arguments, review the Help under Quick License Manager Professional / Http Methods / GetActivationKey

Step by Step Procedure to implement Software Activation

- Define a Product in Quick License Manager.
- Install the QLM web service
- Configure the QLM Console to connect to the QLM web service
- Modify your application so that you can handle the following scenarios:
  - If an evaluation key is entered, you process it without activation.
  - If an activation key is entered, you invoke your web service to get a machine bound key. Once you have received a computer bound key, store it along with the activation key with your application's settings. When your application is started, load these 2 keys. Validate the computer bound key using ValidateLicenseEx. If the key validates properly, enable your software. If it fails, display the activation key and allow the user to activate his license again.
  - If a machine bound key is entered, you validate that it matches the current computer.
  - If a Generic key is entered, you enable your software without activation. Note that even if you implement software activation, you may want to support Generic keys for Corporate customers that purchase site licenses. Software Activation in this situation may not be desirable.
- If you use a third-party e-commerce provider to process your orders, have that provider invoke the GetActivationKey method when an order is placed so that the email the customer receives contains an Activation key that requires to be activated.
- When the user enters the Activation key in your application, you invoke the ActivateKey method providing the computer identifier to the service. A computer bound license key is returned and the software is enabled. The ActivateKey method can be invoked in 2 ways: (a) by using the URL syntax as shown above or (b) if you are using .NET, you can call the ActivateLicense method that is available in the QlmLicenseLib assembly. For more details, see the **Activating a license key** section below.

To invoke GetActivationKey, you need to send an http request as follows:

http://yourserver/qlm/qlmservice.asmx/GetActivationKey?is_productid=1&is_majorversion=1&is_minorversion=0

where is_productid, is_majorversion and is_minorversion are the values defined for the required product in Quick License Manager.

Basic eCommerce Providers

A basic eCommerce provider is a provider that does not allow calling an external script such as the QLM Web service during the purchase process. This type of provider typically allows you to upload a list of license keys that are distributed to buyers upon purchase.

In order to integrate with basic eCommerce providers, follow the steps below:

- Use QLM to generate a set of activation keys that are not bound to any user.
- To create these keys, click on Licenses, and select Create.
- Select your product and the number of license keys to create
- Uncheck the Customer Email checkbox and the Single Activation Key checkbox.
- Specify the Number of Licenses to create
- In the Results dialog, double click on the result.
- Copy the generated keys to the clipboard and paste them in your eCommerce provider's web site.
- In your application, you now need to enter the user information and activate the license key. The ActivateLicenseDialog API can be used to display a form for entering the user contact information as well as the activation key. When the form is submitted, the new user is added to the database and his license is activated.
- Alternatively, you may create your own form to capture contact information. If you do so, you will need to call the following functions:
    - DefineProduct
    - AddUser
    - ActivateLicenseForUser

Ecommerce Providers

If you are using an ecommerce provider to sell your software, QLM provides a mechanism for integrating with your ecommerce provider. The QLM Web service provides a method that can be invoked from your ecommerce provider as follows:

http://yourserver/yourvirtualdirectory/qlmservice.asmx/GetActivationKey?is_productid=<*productID*>&is_majorversion=<*majorVersion*>&is_minorversion=<*minorVersion*>&is_vendor=<*vendor*>&is_features=<*features*>

where,

<*productID*> is the 2 digit product identifier
<*majorVersion*> is the major version of your product
<*minorVersion*> is the minor version of your product
<*vendor*> is the name of the ecommerce provider as specified in QlmProviders.xml
<*features*> are the features to enable in this key

The following additional arguments can also be specified:

<*is_usemultipleactivationskey*> if true, a single activation key is return for a multiple license request.
<*is_userdata1*> user data to associate to the license key
<*is_affiliateid*> affiliate id to associate to the license key

When the GetActivationKey method is invoked, the web service issues an activation key based on the provided url arguments (is_productid, is_majorversion, is_minorversion, is_vendor, is_features).

Since each ecommerce provider uses a different format for its input (the set of fields entered by the customer on the order form), and expects resutls in their own format, a custom solution is required for each provider. QLM provides a framework for dynamically supporting your own ecommerce provider.

To support your own ecommerce provider, you need to develop a DLL containing a single class that inherits from .Qlmsvc.ECommerceProvider. You need to add reference to the QlmCommerceProvider assembly. A sample implementation of this class for the provider RegNow is provided in the samples folder.

Once you have implemented your own provider class, you must update the QlmProviders.xml will details about your new class.

Authentication of the call to GetActivationKey can be performed in 2 ways: (a) if you provide the user name and password in the QlmProviders.xml, QLM will try to match these credentials against the credentials sent by the ecommerce provider or (b) you can override the implementation of AuthenticateUser and implement your own authentication mechanism.

Note that all the methods exposed by the web service cannot be called with a URL except GetActivationKey and ActivateKey. All other web methods implement a secure authentication mechanism that only accepts requests from the QLM Console.

Example:

http://www.mydomain.com/qlm/qlmservice.asmx/GetActivationKey?is_productid=2&is_majorversion=3&is_minorversion=0&vendor=digibuy&is_features=3

The URL above generates an activation key for ProductID = 2, MajorVersion = 3, MinorVersion = 0.
The returned response is customized for Digibuy (e-commerce provider).
The enabled features are: Feature 1 and Feature 2 (1+2=3).

Digibuy

If you are using Digibuy as an ecommerce provider, QLM integrates seamlessly with Digibuy's ordering system.

To have Digibuy invoke QLM during an order, do the following in Digibuy Admin Console:

- Select a Product.
- Click on "reg codes".
- Locate the **http://** field
- Enter the following URL:

http://yourserver/yourvirtualdirectory/qlmservice.asmx/GetActivationKey?is_productid=<*productID*>&is_majorversion=<*majorVersion*>&is_minorversion=<*minorVersion*>&is_vendor=*digibuy*&is_features=<*features*>

where

- is_vendor = Digibuy
- is_productid = your product id as defined in QLM
- is_majorversion = your product's major version as defined in QLM
- is minorversion = your product's minor version as defined in QLM
- is_qlmversion = 5.0.00
- is_features = semi comma separated list of feature sets and ther corresponding values. Example: is_features=0:3;1:0;2:7;3:25
- is_user = username defined in the QlmProviders.xml (optional)
- is_pwd = password defined in the QlmProviders.xml (optional)

Additionally, Digibuy provides a mechanism for protecting this request with a password. On the same page as above, enter a password in the **Password** field.

The password must also be updated in the QlmProviders.xml file located in the bin folder. Edit QlmProviders.xml using any text editor and udpate the user and password fields.

In the user field, specify your Digibuy user id. In the password field, specify the password entered above.

```
<provider>

<name>Digibuy</name>

<class>Qlmsvc.Digibuy</class>

<dll>QlmDigibuy.dll</dll>

<user />

<password />

<passwordEncrypted>false</passwordEncrypted>

 </provider>
```

With the steps above completed, place a test order. When the order is placed, the following will occur:

- A new user will be automatically added to the QLM database based on the information collected during the ordering process.
- An activation key will be created in the QLM database and associated to the user.
- An email will be sent to the customer with the activation key.

FastSpring

If you are using FastSpring as an eCommerce provider, QLM integrates seamlessly with FastSpring's ordering system.

After completing the steps below, when a customer purchases your product from FastSpring, FastSpring will automatically get a license key from the QLM web service and then update the QLM database with the license and customer information.

To integrate FastSpring with QLM, peform the following in the FastSpring Control Panel:

FastSpring Setup for a Basic Product

- Select your product in the FastSpring Control Panel
- Add a Fulfillment Action:
- On the Licenses tab, select "Remote" and click on Next
- Set the URL to: http://quicklicensemanager.com/qlmdemov6/qlmservice.asmx/GetActivationKey?is_vendor=fastspring&is_productid=1&is_majorversion=1&is_minorversion=0&is_qlmversion=5.0.00
- You should update the URL above to point to your server and specify your product ID, major and minor version.
- Method: HTTP POST
- POST Encoding: UTF-8
- License Name Type: Person Full Name + Email Address
- Output Format: Plain Text, Single-Line License
- Click on Create
- Click on Save
- In the Fulfilment Actions section, click on Add
- Select Email / Web Notification and click Next
- Reuse Options: Reusable On Multiple Products
- Subject : #{orderItem.display} - Order #{order.reference}
- Email Text Contents :

```
Dear #{order.customer.fullName},

Thank you for your purchase of #{orderItem.display}.

Your Activation Key is: #{orderItem.fulfillment.license.outcome.licenses.list}
You can download #{orderItem.display} from:
http://soraco.co/products/qlm/qlmsetup5.exe

Should you have any questions please contact us at: sales@soraco.co

Sincerely,
Soraco Technologies
```

- You can customize the Email Html and Web tabs as well.

FastSpring Setup for Updating Contact Information

To transfer the contact information from FastSpring to QLM, follow the steps below:

- From the FastSpring Control Panel Home page, click on the "Notify" icon in the top toolbar.
- Click on Add Notification Rule.
- Format: HTTP Remote Server Call.

- Type: Order Notification.
- Remote Server URL: http://quicklicensemanager.com/qlmdemov6/qlmservice.asmx/UpdateUserInformation
- Click on Next
- In the HTTP Parameters section, click on Add Parameter.
- Name: is_avkey
- Value: #{orderItem.fulfillment.license.outcome.licenses.list}
- Click on Add Parameter again.
- Name: is_vendor
- Value: fastspring
- Optionally, add an is_user and is_pwd parameters to enforce authentication. The user and password can be specified from the QLM Application, under Manager Keys / Commerce Providers.

With the steps above completed, place a test order by clicking on the FastSpring Control Panel Home page, then Store Testing. Once your test order is completed, an activation key will be created in the QLM database along with the corresponding customer and order information.

FastSpring Setup for Maintenance Plan

If you sell a yearly maintenance plan to your customer, you can configure FastSpring to automatically update the maintenance plan expiry date in QLM.

- Click on Products and Pages
- Click on Create Subscription Product
- Name: Yearly Maintenance Plan
- Regular Period Length: 1 year
- USD: xx
- Click on Create
- In the Fulfillment section, click on Add
- Click on the License tab, select Remote and click Next
- URL: http://quicklicensemanager.com/qlmdemov6/qlmservice.asmx/RenewMaintenancePlan?is_vendor=fastspring&is_user=john&is_pwd=fast123
- License Name Type: Person Full Name + Email
- Click on Create
- Click on Save
- Click on the Home button
- Click on Custom Fields
- Click on Create Customer Field Configuration
- Name: custom_referrer (THIS NAME IS FIXED, IT CANNOT BE CHANGED)
- Click on Next
- Display: Activation Key
- Click on Add Form Field
- Input Type: Textbox
- Required: Yes
- Name: custom_referrer
- Question Text: Activation Key
- Question Description: Enter your current activation key (starts with the letter A)
- Click on Add

- Click on Save
- Click on Save
- In the Conditions section, click on Edit
- Select Order Environment Condition and click Next
- Environment Tag Exists: is_maintenance_plan
- Click on Create
- Click on Save
- Click on the Home page

To test the maintenance plan configuration:

- Click on Store Testing
- Click on Optional Parameters
- Tags: is_maintenance_plan
- Click on Testing Links tab then Detail for “demo yearly maintenance plan”
- Click on Order Now, fill contact info then Next
- The next page should show the Activation Key field.
- Go to the QLM Application and locate the license created above.
- Enter the Activation Key in the order form and click on Order
- Once the order is completed, go to QLM and locate your license key
- Refresh the page, select the license and click on the Edit.
- Note how the maintenance plan was increased by a year.

FastSpring Setup for Purchasing an Upgrade

If you sell upgrades to a specific version of your software, you can integrate FastSpring with QLM’s license upgrade feature.

- Create a Product in a way similar to the first section
- Click on Create Product
- Name: Upgrade
- USD: xx
- Click on Create
- In the Fulfillment section, click on Add
- Click on the License tab, select Remote and click Next
- URL: http://quicklicensemanager.com/qlmdemov6/qlmservice.asmx/UpgradeLicense?is_vendor=fastspring&is_productid=1&is_majorversion=2&is_minorversion=0&is_qlmversion=5.0.00&is_user=john&is_pwd=fast123
- License Name Type: Person Full Name + Email
- Click on Create
- Click on Save
- Create an Email Fulfillment similar to the main Product
- Click on the Home button
- Click on Custom Fields
- Click on “custom_referrer”
- In the Conditions section, click on Edit
- In the Active Conditions section, click Edit
- Environment Tag Exists: is_maintenance_plan, is_ugprade
- Click on Save
- Click on the Home page

To test the Product Upgrade configuration:

- Click on Store Testing
- Option Paramaters / Tags: is_upgrade
- Click on testing links: Demo Upgrade / Detail
- Click on Order Now, then Next
- In the QLM Application, create a license key for product Demo 1.0
- Enter this activation key in the FastSpring form
- Click on Compete Order
- Once the order is completed, go to QLM and click on Today's orders
- Note how a new key is created replacing the old one.

Google

If you have a merchant account with Google Checkout and are using Google Checkout as your ecommerce provider, QLM can issue activation keys automatically from Google Checkout's ordering system.

In order to generate license keys for an order and send the keys to a customer automatically at the time of purchase, QLM can be used in conjunction with Google's Checkout Merchant Account Settings. Google Checkout provides a mechanism for immediate notification when a customer purchases your product. QLM integrates with this mechanism using QlmGoogleCheckout. QlmGoogleCheckout is an ASP.Net application included with QLM.

When an order is charged either manually by the merchant or automatically by Google Checkout, QlmGoogleCheckout adds the customer to the QLM database, creates an order in QLM, generates one or more activation keys, and emails the activation keys to both the merchant and the customer. An order can include multiple items. One activation key per item is issued. Each item must correspond to a product defined in QLM.

The Merchant's Shopping Cart

The following must be added to your shopping cart to map an item to its corresponding QLM product.

If using an XML shopping cart add the following to your cart for each item:

```
<merchant-private-item-data>productid=1 major=1
minor=0</merchant-private-item-data>
```

If using an HTML shopping cart add the following to your cart for each item:

```
<input type="hidden"
name="shopping-cart.items.item-1.merchant-private-item-data"
value="productid=1 major=1 minor=0" />
```

- Where productid represents your QLM productID
- Where major represents the major version of your QLM product
- Where minor represents the minor version of your QLM product
- Note that including a second item in an HTML cart would read `shopping-cart.items.item-2....`

The Merchant's Google Checkout Account Settings

The following steps are required to integrate QLM with Google Checkout

- Set up the QLM Web service on a web server by following installation steps found in the QLM Professional Help.
- Define and upload your products to the QLM Web service. Refer to the QLM Help for further information.
- Set up the QLM ASP.Net application **QlmGoogleCheckout** into its own virtual directory. The virtual directory must be configured to require SLL or TLS.
- Configure the QlmGoogleCheckout application's Web.config file. Set the following application settings fields:
    - <appSettings>
        - <add key="company" value="Your Company Name"/>
        - <add key="supportEmail" value="support@yourdomain.com"/>
        - <add key="smtpServer" value="localhost"/>
        - <add key="smtpPort" value="25"/>
        - <add key="communicationEncryptionKey" value="communicationKeyUsedByQLMWeb"/>
        - <add key="qlmVersion" value="5.0.00"/>
        - <add key="webServiceUrl" value="http://yourserver/qlmweb/qlmservice.asmx"/>
        - <add key="merchantID" value="Your Merchant ID"/>
        - <add key="merchantKey" value="Your Merchant Key"/>
    - </appSettings>
- From your Google Checkout Merchant account, set the following settings
    - From the Settings Tab, select the Integration link on the left hand side.
    - Set the API callback URL to the program to QLMGoogleCheckout's GoogleNotify.aspx The URL might look something like http://yourdomain.com/qlmgoogle/GoogleNotify.aspx
    - Set the Callback method to XML

- (Optional) If you wish to capture the buyer's phone number in QLM, select the Advanced Settings and and Check the checkbox next to "Return the buyer's billing phone number in the new order notification.

QLM Order Processing

When a customer purchases your product, Google Checkout sends a new order notification. QLM will at this point create a new order in its own database. If the customer does not already exist, a new customer will be added. The customer's billing coordinates are saved. A QLM order consists of the following fields:

- ActivationKey
- ProductName
- MajorVersion
- MinorVersion
- OrderDate
- ActivationDate
- OrderID
- NumLicenses

Financial information such as the amount of the order is not stored in QLM. When you charge the order from your Google Merchant Account, Google Checkout sends a state change notification. QLM will at this time, send the customer by email his or her activation keys. The merchant is also sent the activation keys by email. All other notifications from Google Checkout are ignored.

Below is a typical example of the email sent by QLM:

Dear customer name,

Please find below additional information regarding your purchase.

Google order number: 17274818052274
Date of purchase: 11/28/2007

Product: your product name1
Quantity: 1
Activation Key: A00F-C858-8FE1-1340-6001-00A0

Product: your product name2
Quantity: 2
Activation Key: A0B5-C078-97FA-1220-4102-00A0

For additional inquiries, please contact support@yourdomain.com with the above information.

Thank you,

Your company name

QLM Google Checkout Deployment

To deploy the QLM Google Checkout module to your server, follow the steps below:

- Create a virtual directory on your IIS server called qlmgoogle.
- Configure the virtual directory to require an SSL connection.
- Copy all the files located in the DeployToServer\QlmGoogleCheckout folder to the qlmgoogle virtual directory.
- Edit the web.config and customize it as described above.

Paypal

If you are using Paypal as an ecommerce provider, QLM integrates seamlessly with Paypal's ordering system. In order to generate a license key for an order and send the key to a customer automatically at the time of purchase, QLM can be used in conjunction with PayPal's Instance Payment Notification (IPN). PayPal's IPN provides immediate notification when a customer purchases your product.

QLM's integration with Paypal performs the following tasks:

1. It validates the paypal request.
2. It contacts the QLM web service and issues an activation.
3. It sends an email to the customer with the activation key.

In order to set up the IPN, you need to modify your PayPal account's Profile to enable Instant Payment Notification and specify a URL to the QLM service that handles Paypal Notifications. Alternatively, you can activate IPN by including the notify_url in your PayPal button HTML.

The URL to the QLM service that handles Paypal notifications is: http://yourserver/qlm/qlmpaypalipn.aspx

For detail instructions on Instant Payment Notification, please refer to PayPal's Order Management Integration Guide found on its website under the tab Merchant Services.

Process Customization

The web.config on the QLM web service contains a specific section for the Paypal integration. The section is located under applicationSettings and is called: QlmPaypalIpn.Properties.Settings.

All the settings in this section should be customized as follows:

- vendorCompanyName: Name of your company.
- vendorCompanyEmail: Your email address. QLM will cc you every time an email is sent to a customer.
- tempalteFile: Name of the template file. The template file contains the body of the email message that is sent to the customer. The template contains variables that will be replaced at runtime. The template file is located in the same folder as qlmpaypalipn.aspx.
- smtpServer: The SMTP server to use when sending emails.
- smtpUser: The credentials of the user that needs to authenticate with the SMTP server.
- smtpPassword: The password of the user that needs to authenticate with the SMTP server.
- smtpPort: The port of the SMTP server.
- paypalUrl: Url of the paypal service. When testing in a sandbox, the URL is typically: https://www.sandbox.paypal.com/cgi-bin/webscr. When going live, change the URL to: https://www.paypal.com/cgi-bin/webscr
- qlmWebServiceUrl = URL to the QLM web service, typically located in the same virtual directory.
- defaultUrlArgs=When configuring your paypal cart or Buy Now page, you typically need to configure the productid, major version and minor version (more details later). If these arguments are not provided, the system will default to the product, major and minor version defined in this setting.
- loggingLevel: specifies the level of logs generated. The levels are defined as follows: Errors: 1 - Warnings: 2 - Information: 4 - Verbose: 8. To log errors and warnings, set the loggingLevel to 3. To log all events, set the loggingLevel to 15.
- paypalFields: PayPal posts variables to the IPN process. Some of these variables are declared in your shopping cart or in your company's purchase page and others are sent by PayPal automatically. For example, your purchase page defines your company by using the variable called "business". Similarly, PayPal uses the variable payer_email to identify the customer's email address. The paypalFields setting lists the paypal fields that QLM will process. A list of the most common fields is preconfigured but you can add more fields if needed. Any field that is added can be used in the email template.

  When configuring your Buy Now button or your Paypal cart, you must configure the following IPN variables. These variables are required by the QlmPaypalIPN process:

  - item_name: product's name, must be changed to be the name of your product
  - item_number or custom: &is_productid=x&is_majorversion=y&is_minorversion=z&is_features=f where x,y,z and f must be replaced with the values that correspond to your product. Additionally, you can customize the email template to use when sending an email to the customer. To customize the email template, add the *is_emailtemplate* argument as follows: &is_emailtemplate=template1.txt. The email template files must be located in the web service folder, in the same location as the default QlmEmailTemplate.txt template file.

Plimus

If you are using Plimus as an ecommerce provider, QLM integrates seamlessly with Plimus's ordering system.

To have Plimus invoke QLM during an order, do the following in Plimus Control Panel:

- Click on My Account
- Locate your Product and click on Setup.
- Locate a contract and click on Setup.
- Click on the **License Keys** link.
- In the License Group/Method field, select: **Custom HTPP request**.
- In the Call Method field, select: **One call per Order**.
- In the URL for HTTP request, enter the following:

  http://yourserver/qlm/qlmservice.asmx/GetActivationKey?is_vendor=plimus&is_productid=1&is_majorversion=1&is_minorversion=0&is_qlmversion=5.0.00&is_features=0&order_id=<INVOICE_ID>&email=<CUSTOMER_EMAIL>&name=<CUSTOMER_NAME>&company=<COMPANY_NAME>&addr1=<CUSTOMER_ADDRESS1>&addr2=<CUSTOMER_ADDRESS2>&city=<CUSTOMER_CITY>&state=<CUSTOMER_STATE>&zip=<CUSTOMER_ZIPCODE>&country=<CUSTOMER_COUNTRY>&phone=<CUSTOMER_PHONE>&quantity=<QUANTITY>

where

- is_vendor = plimus
- is_productid = your product id as defined in QLM
- is_majorversion = your product's major version as defined in QLM
- is minorversion = your product's minor version as defined in QLM
- is_qlmversion = 5.0.00
- is_features = semi comma separated list of feature sets and ther corresponding values. Example: is_features=0:3;1:0;2:7;3:25
- is_user = username defined in the QlmProviders.xml (optional)
- is_pwd = password defined in the QlmProviders.xml (optional)

With the steps above completed, place a test order. When the order is placed, the following will occur:

- A new user will be automatically added to the QLM database based on the information collected during the ordering process.
- An activation key will be created in the QLM database and associated to the user.
- If you would like to customize the email that is sent to the customer with the activation key, in the Plimus control panel, click on the **Order Email** link and add the following line to the body of the email:

  Your Activation Key: <LICENSE_KEYS>

Regnow

If you are using Regnow as an ecommerce provider, QLM integrates seamlessly with Regnow's ordering system.

To have Regnow invoke QLM during an order, do the following in Regnow Control Panel:

- Click on Products
- Select a Product and click on Edit.
- Click on the **Manage Delivery Options** tab.
- Locate the **Method** field and select **Post.**
- Locate the **http://** field and enter the following URL:

http://yourserver/yourvirtualdirectory/qlmservice.asmx/GetActivationKey?is_productid=<*productID*>&is_majorversion=<*majorVersion*>&is_minorversion=<*minorVersion*>&is_vendor=*regnow*&is_features=<*features*>

where

- is_vendor = regnow
- is_productid = your product id as defined in QLM
- is_majorversion = your product's major version as defined in QLM
- is minorversion = your product's minor version as defined in QLM
- is_qlmversion = 5.0.00
- is_features = semi comma separated list of feature sets and ther corresponding values. Example: is_features=0:3;1:0;2:7;3:25
- is_user = username defined in the QlmProviders.xml (optional)
- is_pwd = password defined in the QlmProviders.xml (optional)

With the steps above completed, place a test order. When the order is placed, the following will occur:

- A new user will be automatically added to the QLM database based on the information collected during the ordering process.
- An activation key will be created in the QLM database and associated to the user.
- An email will be sent to the customer with the activation key.

ShareIt

If you are using ShareIt as an e-commerce provider, QLM can be used to issue license keys and save the order and customer information when a customer purchases your product using ShareIt's ordering system.

To have ShareIt invoke QLM at the time of purchase, do the following:

- Setup the QLM web service on your server
- Optionally, configure authentication to connect to ShareIt. To configure authentication, click on Manage Keys / Tools / eCommerce Providers and specify the credentials for ShareIt.
- Create or edit your product in ShareIt's Control Panel
- Ask ShareIt personnel to set your product's **Key Generator URL** to the following URL:

http://yourserver/yourvirtualdirectory/qlmservice.asmx/GetActivationKey?&is_vendor= ShareIt**&is_productid=**<productid>**&is_majorversion=**<majorversion>**&is_minorversion=** <minorversion>**&is_qlmversion=**<qlmversion>**&is_pwd=**<pwd**&is_user=**<username>**&is_features=**

Where:

- is_vendor = ShareIt
- is_productid = your product id as defined in QLM
- is_majorversion = your product's major version as defined in QLM
- is minorversion = your product's minor version as defined in QLM
- is_qlmversion = 5.0.00
- is_features = semi comma separated list of feature sets and ther corresponding values. Example: is_features=0:3;1:0;2:7;3:25
- is_user = username defined in the QlmProviders.xml (optional)
- is_pwd = password defined in the QlmProviders.xml (optional)

With the steps above completed, place a test order. When the order is placed, the following will occur:

- A new user will be automatically added to the QLM database based on the information collected during the ordering process.
- An activation key will be created in the QLM database and associated to the user.
- An email will be sent to the customer with the activation key.

SWREG

If you are using SWREG as an ecommerce provider, QLM integrates seamlessly with SWREG's ordering system.

To have SWREG invoke QLM during an order, do the following in the SWREG Control Panel:

- Click on Create/Edit Products
- Select a Product and click on Edit.
- Click on the **Edit Delivery Method**.
- Click on **Edit Email.**
- Locate the **Keycode generator URL**field and enter the following URL:

http://yourserver/yourvirtualdirectory/qlmservice.asmx/GetActivationKey

When invoking the SWREG order form from your web site, add the following argument to the SWREG URL:
&t=pid%3d<*productID*>%26mj%3d<*majorVersion*>%26mn%3d<*minorVersion*>%26vn%3d*SWREG* %26fe%3d<*features*> *%26ur%3d<user> %26pw%3d<password> &x=1*

where,

- *vn = SWREG*
- *productID = your product id as defined in QLM*
- *majorVersion = your product's major version as defined in QLM*
- *minorVersion = your product's minor version as defined in QLM*
- *features = semi comma separated list of feature sets and ther corresponding values. Example: is_features=0:3;1:0;2:7;3:25*
- *user = username defined in the QlmProviders.xml (optional)*
- *password = password defined in the QlmProviders.xml (optional)*

Note: Due to limitations in the maximum number of characters in SWREG's user_text field, the name of the arguments above have been abbreviated.

For example:

https://usd.swreg.org/cgi-bin/s.cgi?s=46994&p=46994TESTPROD&v=0&d=0&q=1&t=pid%3d4%26mj%3d4%26mn%3d0%26vn%3dSWREG&x=1

With the steps above completed, place a test order. When the order is placed, the following will occur:

- *A new user will be automatically added to the QLM database based on the information collected during the ordering process.*
- *An activation key will be created in the QLM database and associated to the user.*
- *An email will be sent to the customer with the activation key.*

UltraCart

Quick License Manager integrate with UltraCart's ordering system to generate activation codes automatically.

To have UltraCart invoke QLM during an order, do the following:

From within your UltraCart account:

From Item Management -> Items

- Locate your Product and open the Item Editor for this product
- In the Item Editor, select the **Digital Delivery** tab
- In the **Activation Codes** section of the Digital Delivery tab:
- Select **Retrieve Real-time**
- Enter the following **URL** (change the 'yourserver' name to your own)

  http://yourserver/qlm/qlmservice.asmx/GetActivationKey?is_vendor=ultracart&is_productid=1&is_majorversion=1&is_minorversion=0&is_qlmversion=5.0.00&is_features=0

  Where

  - is_vendor = ultracart
  - is_productid = your product id as defined in QLM
  - is_majorversion = your product's major version as defined in QLM
  - is_minorversion = your product's minor version as defined in QLM
  - is_qlmversion = 5.0.00
  - is_features = semicolon separated list of feature sets and ther corresponding values. Example: is_features=0:3;1:0

- Enter a value in the **Shared Secret** field. Remember this value as it needs to be set in QLM as well.

Next configure QLM to include the UltraCart provider as follows:

From Manage Keys-> License Management tab

- Select **Tools -> eCommerce Providers**
- Click the Add button and enter the following values:
  - Name: UltraCart
  - Class: Qlmsvc.UltraCart
  - Dll: UltraCart.dll
  - User:
  - Password: ***Shared Secret*** *entered in UltraCart Activation Code section*

Edit your QLM server's Web.config file and set the following setting:

<setting name="defaultVendor" serializeAs="String">
&ltvalue>UltraCart</value>
</setting>

With the above steps completed, place a test order. When the order is placed, the following will occur:

- A new user will be automatically added to the QLM database based on the information collected during the ordering process.
- An activation key will be created in the QLM database and associated to the user.

Adding a new eCommerce Provider

QLM Professional supports several eCommerce providers. If QLM does not support your eCommerce provider, you can either extend QLM to support your provider or contact our Professional Services to receive a quote for supporting your eCommerce provider.

Creating your eCommerce Provider plugin

The sample program Regnow located in:

%AppData%\My Documents\Quick License Manager\Samples\QLMPro\Regnow
provides a starting point to create your own eCommerce provider plugin.

The eCommerce provider plugin serves 2 purposes: (a) to extract information sent by your eCommerce provider during the purchase process and map this information to QLM fields and (b) to format the output of the license key request sent by your eCommerce provider using the syntax expected by your eCommerce provider.

Configuring data extraction from eCommerce provider into QLM Database

Upon an item purchase, eCommerce providers can typically invoke a web service and provide details about the order. The information provided includes data such as customer name, email address, product ordered, number of lcienses, etc.. In order to capture this information and update the QLM database accordingly, the plugin needs to map the eCommerce provider data into QLM fields. A set of properties can be overwrritten in the plugin class to customize this mapping. The properties that can be overwritten are:

Order ID

OrderStatus

ReceiptID

CustomerEmail

CustomerName

CustomerCompany

CustomerAddress1

CustomerAddress2

CustomerCity

CustomerState

CustomerZip

CustomerCountry

CustomerPhone

CustomerIP

CustomerNotes

Quantity

MaintenancePlan

Configuring the format of the license key

Upon purchase of an item, eCommerce providers typically expect a license key to be returned when they invoke the QLM web service. The format of the returned key or keys differs for each eCommerce provider.

You can customize via the QLM plugin the format of the returned key to meet your eCommerce provider's requirements.

A set of methods can be overwrritten in the plugin class to customize this syntax. The methods that can be

overwritten are:

WriteStart

WriteEnd

WriteKeyStart

WriteKeyEnd

WriteOrderInfo

WriteMaintenanceRenewalResponse

WriteUpgradeLicenseResponse

WriteError

Configuring authentication

eCommerce providers connect to the QLM Web service via an http request. To ensure that hackers do not contact your web service easily, you can provide a username / password on the url command line. Since the web service is invoked directly from your eCommerce provider, end users will never be able to intercept that URL and discover the password.

The default behavior of QLM supports validating a user/password by comparing the command line arguments with a user/password that you can configure and define in the QLM Console. You can also customize the authentication mechansim by overwritnig the following method:

AuthenticateUser

Deployment of your plugin

Once you have completed development of your plugin, follow the procedure below to install it and register it:

- Deploy the DLL to the bin folder on your web server, in the same folder as the ***QlmCommerceProvider.dll*** file.
- Launch the QLM Console
- Click on Manage Keys / Tools / eCommerce Providers
- Click on the Add button and enter the data associated with your provider
- Fill in all the fields. For help on each field, click on the text field and read the help in the loght bulb section
- Click OK

Testing your plugin

Once the above steps are completed, you can test your plugin. Most eCommerce providers support a test mode where you can submit a dummy order. To help diagnose problems with your plugin, configure the QLM Web service to log verbose events by updating the **loggingLevel** setting to **15**.

When a test is performed from your eCommerce provider, you can view details about the request from the QLM Events Viewer. To launch the Events Viewer, click on Manage Keys / Tools / View Server Event Log.

Check for Updates

QLM provides a framework that allows you to implement a "Check for Updates" feature for your software. The "Check for Updates" feature allows you to automatically inform your users when a new version of your software is available to download.

Below are the steps required to implement a Check for Updates feature in your application:

- In the Define Product screen, click on the Latest Version tab and specify the details of the latest version of your software:
  - Latest Version: Enter the version number of your latest release. Example: 3.1.0
  - URL the latest version: Enter a URL from where the user can download the latest version. This URL can be used from your application to either automatically download your application or simply display the URL to the user.
  - Notes about latest version: Enter notes that you would like to display to the user if QLM detects that a new version is available.
- In QLM, click on Manage Keys / Sites. In the Web Services tab, select the appropriate profile and click on "Upload products to web service" to upload your product data to the server.
- In your application, create a button or a timer based routine that calls the GetLatestVersion function (see API Reference for details).
- Compare the server's version with the installed version and prompt users with the option to upgrade their version if appropriate.

The QLM Web Sample shows how to implement the Check for Updates feature in your application. The sample is located in:

- %AppData%\My Documents\Quick License Manager\Samples\qlmpro\DotNet\Basic\C#\vs2005\QlmCheckForUpdates

Terminal Server

QLM Professional can limit the number of instances of your application running on a Terminal Server.

To restrict the number of instances on a Terminal Server, you must set the "Floating Seats" property when creating an Activation Key.

Below are the steps required to specify the number of allowed terminal server instances:

- Click on the Manage Keys tab.
- Click on the Create button.
- Specify the Floating Seats property to the number of instances allowed to run on the Terminal Server.
- Click OK

In your code, you must also set the LimitTerminalServerInstances property to *true*. The default value is *false*.

Note that an instance is uniquely identified by a session id and a user id. For example, if only 1 instance is allowed, the same user can still launch your application multiple times within the same Terminal Server session. That same user will not be able to start another Terminal Server session and start another instance of your application.

Scheduled Tasks

QLM can schedule and execute tasks such as emailing notifications to your customers or displaying alerts when the QLM database is updated. QLM includes one built-in email scheduled task designed for implementing a Maintenance Plan for your product and several alert type tasks. You can use these tasks as is, customize them to suit your needs, or create other scheduled tasks that meet your special needs.

Note that in order for tasks to run, the Quick License Manager Agent must be running.

Below are the steps required to create scheduled tasks:

- Click on the Manage Keys tab.
- Click on the Scheduled Tasks button.
- Click on Add.
- If you have multiple web servers, select the web service profile to use. Otherwise, select the Default profile.
- Select the Search to execute. The scheduled task will be executed on each record returned by the search.
- Specify how often the task will run.
- To configure an Email notification task, click on the Email tab and check the Enable Email Notification box, then fill in all the remaining fields.
- To configure an Alert notification task, click on the Alert tab and check the Enable Alert box, then fill in all the remaining fields. The Message fields supports variables such as %FullName% or %ActivationKey%.

Note that QLM uses your Outlook 2003 or 2007 client to send emails. You need to be logged in to the system for email notifications to work.

Maintenance Plan

QLM provides a complete framework for implementing a maintenance plan for your software.

When a customer activates a license key, the QLM Web service validates that the license key corresponds to the version the customer is running. For example, if a customer is running version 6.0 of your software but has an Activation key for version 5.0, the QLM Web service will fail to activate the license.

However, if a customer purchases a maintenance plan for your software, the QLM Web service will allow activation of a recent version of your software with an activation key for a previous version. For example, if a customer is running version 6.0 of your software but has an Activation key for version 5.0, the QLM Web service will successfully activate the license if the maintenance plan is active and the MaintenancePlanRenewal date is greater than the Release Date of the new version of your product. The Release Date of your product can be defined on the Define Product page in the QLM Console.

The QLM database stores maintenance plan information in the MaintenanceRenewalDate field. If the maintenance plan is not enabled, this field will be empty. If the maintenance plan is enabled, the field will be set to the expiry date of the maintenance plan.

The MaintenanceRenewalDate can be set during the order process or manually via the QLM console.

To enable the maintenance plan during the ordering process, you can invoke the following URL:

http://yourserver/qlm/qlmservice.asmx/EnableMaintenancePlan?is_vendor=<vendor>&is_user=<user>&is_pwd=<pwd>

where:

- <vendor> is the name of your vendor (see list of supported eCommerce providers)
- <user> is a user to use for authentication (as defined in the QlmProviders.xml file)
- <pwd> is a password to use for authentication (as defined in the QlmProviders.xml file)

The maintenance plan date is set to 365 days after the Order Date. You can control this setting from the web.config file of the QLM web service.

Alternatively, you can customize your eCommerce provider order form to publish a new field during the POST to specify whether the maintenance plan was purchased. The name of the field is typically different for each eCommerce provider. The table below describes the name of this field for each of the supported eCommerce providers:

Provider	Field Name
Digibuy	maintenance
Regnow	yearly_maintenance_plan
Paypal	MaintenancePlan
ShareIt	yearly_maintenance_plan
Plimus	is_maintenance_plan

To set the maintenance plan using the QLM console:

- For a new activation key, click on Manage Keys / Licenses / Create and check the Maintenance Plan check box.
- For an existing activation key, click on Manage Keys / Licenses / Edit then specify a Renewal Date for the Maintenance Plan.

Maintenance Plan Email Notification

To remind customers to renew the maintenance plan, you can schedule a task that will send your customers a reminder email prior to the expiry of the maintenance plan. For more details, see the help on Scheduled Tasks.

Place an order to renew a maintenance plan

When a maintenance plan expires, you can renew the maintenance plan from your eCommerce provider's ordering system by invoking the following URL:
http://yourserver/qlm/qlmservice.asmx/RenewMaintenancePlan?&is_avkey=<activationKey>&is_vendor=<vendor>

where:

- <activationKey> is the activation key of the customer
- <vendor> is the name of your vendor (see list of supported eCommerce providers)

Subscription licensing

If you sell your software as a subscription, QLM can create license keys that expire at a specific date or after a set duration. This is identical to trial keys.

To create a license key with an expiry date, you can use one of the following methods:

1. The QLM Console (Manage Keys / Create)
2. The QLM API (CreateActivationKeyWithExpiryDateEx).
3. The Http method GetActivationKeyWithExpiryDate, typically invoked directly from your ecommerce provider. See Http methods in the help for more details.

Once a subscription expires, if your customer does not renew the subscription, the license expires and your software no longer runs.

If your customer decides to renew the subscription, you can renew the license without sending your customer a new license key. The customer simply needs to reactivate their license to extend it to the new expiry date.

Extending a subscription can be performed in 3 different ways:

1. From the QLM Console, under Manage Keys / Renew Subscription
2. By using the QLM API RenewSubscription from your application.
3. By using the Http method RenewSubscriptionHttp from your ecommerce provider. See Http methods in the help for more details.

Affiliates

If you use affiliates to sell your product, QLM allows you to manage your affiliates' sales. QLM provides a web portal for your affiliates to generate license keys for their customers. You can configure the maximum number of license keys that an affiliate can generate per computer or overall. The Affiliates Portal site generates Activation keys or Computer bound keys. Note that the Affiliates Portal is not a replacement for the QLM Application. It provides a subset of the features available in the QLM Application / Manage Keys tool.

The Affiliates portal is an add-on to QLM. To register your license and enable the Affiliates Portal add-on:

- Start the QLM Application
- Go to the Manage Keys tab
- Click on Sites
- Select your web service profile
- In the Portal License Key field, enter your QLM Affiliates Portal license key and click on Register. Note that the QLM Portal License Key is not the same as your QLM Professional or Enterprise license key. If you have not purchased the QLM Affiliates Portal add-on, you can do so from our web site.

To define an affiliate, click on Tools / Affiliates and add your affiliates.

The Affiliates web portal can be installed via the setup (qlmwebsvcsetup.exe) or manually.

If you can run a setup program on your web server, run the qlmwebsvcsetup.exe on your server and make sure that the Affiliates Portal feature is selected.

If you cannot run a setup program on your web server, following are the manual steps to install the Affiliates Portal:

- Create a virtual directory on your web server called: QlmAspClient
- Upload all the files in the DeployToServer\QlmAspClient folder to the QlmAspClient virtual directory
- Enable ASP.NET 2.0/3.5 for the virtual directory.
- Customize the following appSettings in the web.config file as follows:
  - communicationEncryptionKey
  - qlmVersion
  - webServiceUrl
  - statesFile
  - countriesFile
  - logo
- Additionally, you should customize the connectionStrings settings in the web.config file.

The Affiliates portal uses ASP.NET forms authentication to validate users. To enable ASP.NET forms authentication, the QLM database needs to be updated to maintain authentication information. To upgrade your QLM database, run the sql2005.aspnet.sql script located in the following folder (if you installed the Affiliates Portal via the qlmwebsvcsetup.exe, this step can be skipped):

C:\Program Files\Soraco\QuickLicenseMgr\DeployToServer\QlmWebService\Db

To access the Affiliates Portal, enter the following URL in a browser:

- http://yourserver/qlmaspclient/qlmaspclient.aspx

The first step in setting up a portal is to create the Administrator's user account. To do so, start the QLM Application and:

- Go to the Manage Keys tab.
- Click on User Accounts in the Cloud & Affiliates group.
- Click on the Add button to create a new user account.
- Fill in all the fields as required. When asked to associate an Affiliate with the user account, if you select None, the user will be able to create an unlimited amount of keys. If you select an affiliate, the user will be able to create as many keys as allowed for this specific Affiliate.

Now that you have an account, you may login to the portal at the following URL:

http://yourserver/qlmaspclient/qlmaspclient.aspx

Once logged in, you can create license keys for your products as well as add new customers.

To create user account for your affiliates:

- Go to the Manage Keys tab.
- Click on Affiliates in the Cloud & Affiliates group.
- Click on the Add button to create a new user affiliate.
- Once the affiliate created, click on User Accounts in the Cloud & Affiliates group.
- Click on the Add button to create a new user account for your affiliate and select the corresponding

Affiliate from the Affiliate dropdown.

- Provide your affiliate with the URL to the portal and his account information.

Server Event Log

When errors occur in the QLM Web service, QLM logs errors messages to the QLM database. These error messages can be viewed from the QLM Console by clicking on Mange Keys / Tools / View Server Event Log. QLM has several levels of logging. QLM can log errors, warnings and information. You can control the logging level by editing the loggingLevel setting in web.config file of the QLM Web service. The possible values for the logging level are:

- Error=1
- Warning=2
- Information=4
- Verbose=8

These settings can be combined together as follows:

To log errors and warnings, set the value to: 3
To log errors, warnings and information, set the value to: 7
To log errors, warnings, information and verbose messages, set the value to: 15

Illegal Computers

QLM can track illegal computers that connect to the web service and logs information about these computers in the database. An illegal computer is defined as a computer that has a valid license key but whose license key is (a) not in the database or (b) in the database but registered for another computer. This situation can occur if a user with a valid license key requests that his license be released from computer A claiming to have uninstalled your program from computer A. If the user subsequently attempts to connect to the web service via computer A, QLM detects this computer as an illegal computer and logs it in the database.

There are two ways to enable illegal computers detection in your application:

- The QlmAspLicenseSite portal provides a web page that your application can connect to by invoking a url. For example:

  http://yourserver/qlmasplicensesite/qlmlicenseinfo.aspx/?is_avkey=AGGI0U0Q00NSSUYY8EH31U1TZ4&is_cpkey=UAJD0M0600PBIUU28NKH1A12JM&is_cpid=MYPC&is_cpname=MYPC&is_qlmversion=5.0.00

  where:

  is_avkey specifies the user's activation key

  is_cpkey specifies the user's computer bound key

  is_cpid specifies the unique computer identifier

  is_cpname specifies the computer name

  is_qlmversion specifies the version of the QLM engine

  When your application is launched, connect to the page above to display license information to your end user. In addition to providing licensing information to the user, this page will detect illegal computers and log them in the database. The QlmAspLicenseSite portal can be found in the DeployToServer folder. The deployment procedure for this portal is identical to the one for the QLM web service. If you have installed the QLM web service using the setup program qlmwebsvcsetup.exe, then the QlmAspLicenseSite is already deployed on your server.

- The QLM API (in QlmLicenseLib.dll) includes a new method called *IsIllegalComputer* to detect illegal computers. This method should be ideally be called every time your application is launched.

To view illegal computers, click on the Manage Keys tab then select the Illegal Computers button.

Note that QLM does not prevent users from running your application if an illegal computer is detected.

Calling the QLM .NET API from VB or C++

The QLM API that connects to the QLM web service was developed with .NET. If your application was developed in C++, VB6 or any other unmanaged code that cannot call .NET API directly, you may still use the QLM API via the COM interface.

To enable the QLM API to be called as a COM interface, you need to do the following:

- **Generate a type library to be refrenced by your code**
  regasm /tlb "<fullpath>\QlmLicenseLib.dll"
- **Register the QlmLicenseLib.dll as a COM object**
  regasm /codebase "<fullpath>\QlmLicenseLib.dll"

Distribute your application

The sections below describe the options to include the QLM required DLLs in your application.

Select the most appropriate option based on the type of setup that you use to deploy your application.

For example, if you use a tool such as InstallShield to install your application, you should use the provided Merge Modules.

Special C

Using the QLM .NET API from VB6 or unmanaged C++

If you have included a reference to QlmLicenseLib.dll in your VB6, unmanaged C++ or any other non .NET based application, your setup must perform the following operations:

32 bit applications

- **Generate a type library to be refrenced by your code**
  %windir%\Microsoft.NET\Framework\v2.0.50727\regasm.exe /tlb "<fullpath>\QlmLicenseLib.dll"
- **Register the QlmLicenseLib.dll as a COM object**
  %windir%\Microsoft.NET\Framework\v2.0.50727\regasm.exe /codebase "<fullpath>\QlmLicenseLib.dll"

64 bit applications

- **Generate a type library to be refrenced by your code**
  "%windir%\Microsoft.NET\Framework64\v2.0.50727\regasm.exe" /tlb "<fullpath>\QlmLicenseLib.dll"
- **Register the QlmLicenseLib.dll as a COM object**
  "%windir%\Microsoft.NET\Framework64\v2.0.50727\regasm.exe" /codebase "<fullpath>\QlmLicenseLib.dll"

Distribute your application using Merge Modules

InstallShield and other MSI based installation tools

If you are using Windows Installer to distribute your application, you can use the merge modules found in the Quick License Manager **redistrib\MergeModules** folder. When including the merge module, set the destination folder of the merge module to be your application's installation folder.

Application Type	Merge Modules to include
Windows Forms .NET 2.0 or higher	Soraco Quick License Manager 5.0 (IsLicense50.msm) Soraco Quick License Manager 5.0 for .NET 2.0 (QlmLicense.net2.msm)
ASP.NET 2.0 or higher Excel 2003 or higher MS-Access 2003 or higher Outlook Add-in VB6	Soraco Quick License Manager 5.0 for .NET 2.0 (QlmLicense.net2.emb.msm)
C++	Soraco Quick License Manager 5.0 (IsLicense50.msm) Soraco Quick License Manager 5.0 for .NET 2.0 (QlmLicense.net2.com.msm)

Note that if your application does not target x64 bit platforms, i.e. your application runs as a 32 bit application on 64 bit machines, then you should include the IsLicense50_x86.msm instead of IsLicense50.msm.

The IsLicense50.msm merge module includes 2 files:

- FileName: *IsLicense50.dll*
- Destination Folder: *[INSTALLDIR]*
- ComponentCode: *{CD707E5F-495E-48E5-8360-EAB26AFC186A}*
- Shared: *No*
- COM Extract at Build or Self-Register: *No*
- Permanent: *No*
- Architecture: *x86*
- FileName: *IsLicense50.dll*
- Destination Folder: *[INSTALLDIR]*
- ComponentCode: *{752E6A64-1248-46B5-87E5-8039A54F6288}*
- Shared: *No*
- COM Extract at Build or Self-Register: *No*
- Permanent: *No*
- Architecture: *x64*

The QlmLicenseLib.net2.msm merge module include several files:

- FileName: *QlmLicenseLib.dll*
- Destination Folder:*[INSTALLDIR]*
- ComponentCode: *{84A850A3-7CDE-4E23-B31E-58C4C716E54F}*
- Shared: *No*
- COM Extract at Build or Self-Register: *No*
- Permanent: *No*
- FileName: *QlmLicenseLib.resources.dll for Spanish*
- Destination Folder: *[INSTALLDIR]es*
- ComponentCode: *{560F660B-2649-45A9-BB53-D499DE808F34}*
- Shared: *No*
- COM Extract at Build or Self-Register: *No*
- Permanent: *No*
- FileName: *QlmLicenseLib.resources.dll for French*
- Destination Folder: *[INSTALLDIR]fr*
- ComponentCode: *{DC0FB90A-AE8A-4261-A82E-BC18B7DFB4C7}*
- Shared: *No*
- COM Extract at Build or Self-Register: *No*
- Permanent: *No*
- FileName: *QlmLicenseLib.resources.dll for Italian*

- Destination Folder: *[INSTALLDIR]it*
- ComponentCode: *{F040D81F-AC78-4CC9-9BD7-18B4F881F34F}*
- Shared: *No*
- COM Extract at Build or Self-Register: *No*
- Permanent: *No*&

- FileName: *QlmLicenseLib.resources.dll for German*
- Destination Folder: *[INSTALLDIR]de*
- ComponentCode: {7C02CB03-FB75-4B80-95C9-0F6421E578C1}
- Shared: *No*
- COM Extract at Build or Self-Register: *No*
- Permanent: *No*

Distribute your application using a Visual Studio Deployment Project

Visual Studio .Net Deployment Project

To add Quick License Manager to your Visual Studio .Net deployment project, follow the steps below:

- Create a deployment project
- Add your project's output in the File System Editor
- This should detect the Quick License Manager dependencies and include the following files:
    - IsLicense50.dll - If you are targeting x64 bit systems, you will need to conditionally install the proper IsLicense50.dll depending on the target platform.
      The x64 bit version of IsLicense50.dll is located in the Redistrib\x64 folder.
    - QlmLicenceLib.dll
    - QlmControls.dll
    - *es\QlmLicenseLib.resources.dll*
    - *fr\QlmLicenseLib.resources.dll*
    - *it\QlmLicenseLib.resources.dll*
    - *de\QlmLicenseLib.resources.dll*

Application Type	Files to include
Windows Forms .NET 2.0 or higher	redistrib\x86\IsLicense50.dll redistrib\x64\IsLicense50.dll redistrib\.net2.0\QlmLicenseLib.dll redistrib\.net2.0\QlmControls.dll
ASP.NET 2.0 or higher Excel 2003 or higher MS-Access 2003 or higher Outlook Add-in VB6	redistrib\.net2.0\QlmLicenseLibEmb\QlmLicenseLib.dll redistrib\.net2.0\QlmControls.dll
C++	redistrib\x86\IsLicense50.dll redistrib\x64\IsLicense50.dll redistrib\.net2.0\QlmLicenseLib.dll redistrib\.net2.0\QlmControls.dll

Distribute your application using a manual procedure

Manual steps to install Quick License Manager files

If you are not using Windows Installer to distribute your application, the following files must be included in your application:

IsLicense50.dll: not shared, to be installed in the same folder as your application

QlmLicenseLib.dll: not shared, to be installed in the same folder as your application

QlmControls.dll: not shared, to be installed in the same folder as your application

If you are targeting x64 bit systems, you will need to conditionally install the proper IsLicense50.dll depending on the target platform.
The x64 bit version of IsLicense50.dll is located in the Redistrib\x64 folder.

Application Type	Files to include
Windows Forms .NET 2.0 or higher	redistrib\x86\IsLicense50.dll redistrib\x64\IsLicense50.dll redistrib\.net2.0\QlmLicenseLib.dll redistrib\.net2.0\QlmControls.dll
ASP.NET 2.0 or higher Excel 2003 or higher MS-Access 2003 or higher Outlook Add-in VB6	redistrib\.net2.0\QlmLicenseLibEmb\QlmLicenseLib.dll redistrib\.net2.0\QlmControls.dll
C++	redistrib\x86\IsLicense50.dll redistrib\x64\IsLicense50.dll redistrib\.net2.0\QlmLicenseLib.dll redistrib\.net2.0\QlmControls.dll

Distribute your VC++ 6.0 or VB 6.0 application

InstallShield and other MSI based installation tools

If your application is developed with a programming language that cannot call the .NET API directly (VB6, Unmanaged C++ code, etc), then you can use the COM method to call the QLM API. In this case, the merge modules to include in your setup are described below.

If you are using Windows Installer to distribute your application, simply include the QlmLicense.net2.com.msm and IsLicense50.com.msm merge modules found in the Quick License Manager installation folder. When including the merge module, set the merge module's destination to be <Program Files>\<Common Files>\Soraco.

Note that if your application does not target x64 bit platforms, i.e. your application runs as a 32 bit application on 64 bit machines, then you should include the IsLicense50_x86.com.msm instead.

In summary, the merge modules to include when targeting both 32 bit and 64 bit systems are:

- Soraco Quick License Manager 5.0 - COM Version(IsLicense50.com.msm)
- Soraco Quick License Manager 5.0 for .NET 2.0 with COM support (QlmLicense.net2.com.msm)

The merge modules to include when targeting 32 bit systems only:

- Soraco Quick License Manager 5.0 - COM Version for x86 (IsLicense50_x86.com.msm)
- Soraco Quick License Manager 5.0 for .NET 2.0 with COM support (QlmLicense.net2.msm)

The IsLicense50.msm merge module includes 2 files:

- FileName: *IsLicense50.dll*
- Destination Folder: *[INSTALLDIR]*
- ComponentCode: *{CD707E5F-495E-48E5-8360-EAB26AFC186A}*
- Shared: *No*
- COM Extract at Build or Self-Register: *No*
- Permanent: *No*
- Architecture: *x86*
- FileName: *IsLicense50.dll*
- Destination Folder: *[INSTALLDIR]*
- ComponentCode: *{752E6A64-1248-46B5-87E5-8039A54F6288}*
- Shared: *No*
- COM Extract at Build or Self-Register: *No*
- Permanent: *No*
- Architecture: *x64*

The QlmLicenseLib.net2.msm merge module include several files:

- FileName: *QlmLicenseLib.dll*
- Destination Folder:*[INSTALLDIR]*
- ComponentCode: *{84A850A3-7CDE-4E23-B31E-58C4C716E54F}*
- Shared: *No*
- COM Extract at Build or Self-Register: *No*
- Permanent: *No*
- FileName: *QlmLicenseLib.tlb*
- Destination Folder: *[INSTALLDIR]*
- ComponentCode: *{84A850A3-7CDE-4E23-B31E-58C4C716E54F}*
- Shared: *No*
- COM Extract at Build or Self-Register: *Yes*
- Permanent: *No*
- FileName: *QlmLicenseLib.resources.dll for Spanish*
- Destination Folder: *[INSTALLDIR]es-ES*
- ComponentCode: *{560F660B-2649-45A9-BB53-D499DE808F34}*
- Shared: *No*
- COM Extract at Build or Self-Register: *No*
- Permanent: *No*
- FileName: *QlmLicenseLib.resources.dll for French*
- Destination Folder: *[INSTALLDIR]fr-FR*
- ComponentCode: *{DC0FB90A-AE8A-4261-A82E-BC18B7DFB4C7}*

- Shared: *No*
- COM Extract at Build or Self-Register: *No*
- Permanent: *No*
- FileName: *QlmLicenseLib.resources.dll for Italian*
- Destination Folder: *[INSTALLDIR]it-IT*
- ComponentCode: *{F040D81F-AC78-4CC9-9BD7-18B4F881F34F}*
- Shared: *No*
- COM Extract at Build or Self-Register: *No*
- Permanent: *No*

Manual steps to install Quick License Manager files

If you are not using Windows Installer to distribute your application, you need to include the following files:
ead" dir="ltr" style="MARGIN-RIGHT: 0px">

IsLicense50.dll: not shared, to be installed in the same folder as your application (INSTALLDIR).

If you are targeting x64 bit systems, you will need to conditionally install the proper IsLicense50.dll depending on the target platform.
The x64 bit version of IsLicense50.dll is located in the Redistrib\x64 folder.

QlmLicenseLib.dll: not shared, to be installed in INSTALLDIR.

QlmLicenseLib.tlb: not shared, to be installed in INSTALLDIR.

es-ES\QlmLicenseLib.resources.dll : not shared, to be installed in INSTALLDIR\es-ES.

fr-FR\QlmLicenseLib.resources.dll : not shared, to be installed in INSTALLDIR\fr-FR.

it-IT\QlmLicenseLib.resources.dll : not shared, to be installed in INSTALLDIR\it-IT.

In addition, you need to register the QlmLicense.dll as follows: regasm /codebase "<fullpath>\QlmLicenseLib.dll"

Localization

QLM .NET API calls may return messages relating the status of a license key validation. By default, all messages returned by the QLM .NET API are in English.

The QLM .NET API also supports French, Spanish, Italian and German messages. To configure your application to display the proper message depending on the language of your customer's system, you need to:

- Locate the following folders in the Quick License Manager folder: es, fr, it, de
- When you deploy your application, copy these folders to the same location as the QlmLicenseLib.dll
- If your customer's system is running Spanish OS, the appropriate resources will be automatically loaded.

Alternatively, if you would like to force a specific language, you need to add the following call to your application prior to initializing any UI:

System.Threading.Thread.CurrentThread.CurrentUICulture = new System.Globalization.CultureInfo("es-ES");

64 bit Support

When distributing your application, depending on the QLM features that you are using, you may need to distribute the following DLLs with your application:

- IsLicense50.dll
- QlmLicenseLib.dll
- QlmControls.dll

IsLicense50.dll is a C++ DLL. If you are targeting a 64 bit platform, you need to ship with your application the 64 bit version of this DLL. The 64 bit version of the DLL is found in the x64 folder.

The *QlmLicenseLib.dll and QlmControls.dll* are .NET assemblies. The same version of these DLLs will work on 32bit or 64 bit platforms.

If your application is intended to run as 32 bit application on a x64 bit operating system, then you need to use the 32 bit version of IsLicense50.dll. If you are using the provided merge modules to integrate QLM into your setup, you should use the IsLicense50_x86.msm or IsLicense_x86.com.msm. These merge modules install the 32 bit version of the IsLicense50.dll regardless of the target platform.

Troubleshooting web service

If you have errors connecting to the web service, check the following troubleshooting tips:

- Try to connect to the web service from the browse by typing the following URL in the browser: http://server/qlm/qlmservice.asmx
- Determine the credentials of the user running the virtual directory. If you have installed the QLM Web service using the qlmwebsvcsetup.exe, an application pool was created called Quick License Manager. The application pool runs by default with the credentials of the NETWORK SERVICE account. The NETWORK SERVICE account will therefore need Modify permissions on the following folders:
    - C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727\Temporary ASP.NET Files
    - C:\Windows\Temp
    - C:\Program Files\Soraco\QuickLicenseMgr\QlmWebSvc\Db

  If you have installed the QLM Web service manually and have not created an application pool, make sure that the anonymous user (IUSR_XXX and IWAM_XXX) have Modify permissions on the following folders:
    - C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727\Temporary ASP.NET Files
    - C:\Windows\Temp
    - C:\Program Files\Soraco\QuickLicenseMgr\QlmWebSvc\Db (or wherever you have installed the database file).
- If you are using MS-Access as a database, verify that the qlm.mdb file on the web server is not read-only. Additionally, verify that the user configured in the IIS QLM virtual directory has Write privileges on the folder where the qlm.mdb file is located. If you are using an Application Pool (recommended), verify that the user who is defined in the Identity tab of the Application Pool has Write privileges on the folder where qlm.mdb is located.
- Verify that the communicationEncryptionKey between the client and the web service match. In the web service, edit the web.config and note the value of communicationEncryptionKey. In the QLM Console, click on Manage Keys, select Tools / Options, click on the Web Service tab and locate the Communication Encyption Key field. Verify that both values match.

Web Service Application API

The functions listed in this section are functions that you need to use within your application in order to interface with the QLM Web Service.

Before calling any of the functions below, you need to set the communicationEncryptionKey property of the QlmLicense object to the value specified in your Site Properties (Manage Keys tab / Site). If you are developing an application in .NET, it is highly recommeded that you obfuscate your code, and more specifically that you encrypt sensitive strings such as the communicationEncryptionKey.

The QLM Web Service provides a more extended set of APIs that you may want to use for managing your license keys or for integrating licensing with any other internal process that you may have. The extended set of functions is described under the **Web Service Management API** section.

ActivateLicense

Activates a license key over the internet. To use a proxy server, you must set the UseProxyServer, ProxyUser, ProxyDomain and ProxyPassword properties prior to calling this function.

C#: void ActivateLicense (string webServiceUrl, string activationKey, string computerID, string computerName, string qlmVersion, string userData1, out string response)

Parameters

webServiceUrl - URL to the QLM Web service.

activationKey - the license key to activate

computerID - the unique computer identifier

computerName - the name of the computer, not required but recommended.

qlmVersion - the version of the QLM Engine

userData1 - User data to associate with the license key

response - XML fragment containing the result of the call. The Xml fragment schema is as follows:

```
<?xml version='1.0' encoding='UTF-8'?>
<QuickLicenseManager>
<pckey>C06C4C90A497F091C2F080501000C076A0578E</pckey>
<userCompany>My Company</userCompany>
<userFullName>John Smith</userFullName>
<userEmail>john@smith.com</userEmail>
</QuickLicenseManager>
```

In the event of an error, the XML fragments returns:

```
<?xml version='1.0' encoding='UTF-8'?>
<QuickLicenseManager>
<error>Details about the error</error>
</QuickLicenseManager>
```

ActivateLicenseEx

Activates a license key over the internet. To use a proxy server, you must set the UseProxyServer, ProxyUser, ProxyDomain and ProxyPassword properties prior to calling this function.

C#: void ActivateLicenseEx (string webServiceUrl, string activationKey, string computerID, string computerName, string qlmVersion, string userData1, string affiliateID, out string response)

Parameters

webServiceUrl - URL to the QLM Web service.

activationKey - the license key to activate

computerID - the unique computer identifier

computerName - the name of the computer, not required but recommended.

qlmVersion - the version of the QLM Engine

userData1 - User data to associate with the license key

affiliateID - ID of affiliate

response - XML fragment containing the result of the call. The Xml fragment schema is as follows:

```
<?xml version='1.0' encoding='UTF-8'?>
<QuickLicenseManager>
<pckey>C06C4C90A497F091C2F080501000C076A0578E</pckey>
<userCompany>My Company</userCompany>
<userFullName>John Smith</userFullName>
<userEmail>john@smith.com</userEmail>
</QuickLicenseManager>
```

In the event of an error, the XML fragments returns:

```
<?xml version='1.0' encoding='UTF-8'?>
<QuickLicenseManager>
<error>Details about the error</error>
</QuickLicenseManager>
```

ActivateLicenseDialog

Creates a form for the user to enter his contact information and the activation key. When submitted, the user is added to the database and his license is activated.

C#: public bool ActivateLicenseDialog(string webServiceUrl, string computerID, string userData1, Control parentWindow, out string response)

Parameters

webServiceUrl - URL to the QLM Web service.

computerID - Unique identifier for the computer

userData1 - user data to associate to this license

parentWindow - parent of the activation form window

response - XML fragment containing the result of the call. The Xml fragment schema is as follows:

```
<?xml version='1.0' encoding='UTF-8'?>
<QuickLicenseManager>
<pckey>C06C4C90A497F091C2F080501000C076A0578E</pckey>
<userCompany>My Company</userCompany>
<userFullName>John Smith</userFullName>
<userEmail>john@smith.com</userEmail>
</QuickLicenseManager>
```

In the event of an error, the XML fragments returns:

```
<?xml version='1.0' encoding='UTF-8'?>
<QuickLicenseManager>
<error>Details about the error</error>
</QuickLicenseManager>
```

ActivateLicenseForUser

Activates a license key over the internet, binds it to a specific user and returns a computer bound license key.Releases a license key over the internet. To use a proxy server, you must set the UseProxyServer, ProxyUser, ProxyDomain and ProxyPassword properties prior to calling this function.

C#: ActivateLicenseForUser(string webServiceUrl, string activationKey, string email, string computerID, string computerName, string qlmVersion, string userData1, string affiliateID, out string response)

Parameters

webServiceUrl - URL to the QLM web service

activationKey - Activation key

email - Email address of user that owns the license

computerID - Unique computer identfier

computerName - Name of computer

qlmVersion - Version of the QLM Engine to use

userData1 - User data to associate with the license key

affiliateID - ID of affiliate

response - XML fragment containing the result of the call. The Xml fragment schema is as follows:

```
<?xml version='1.0' encoding='UTF-8'?>
<QuickLicenseManager>
<pckey>C06C4C90A497F091C2F080501000C076A0578E</pckey>
<userCompany>My Company</userCompany>
<userFullName>John Smith</userFullName>
<userEmail>john@smith.com</userEmail>
</QuickLicenseManager>
```

In the event of an error, the XML fragments returns:

```
<?xml version='1.0' encoding='UTF-8'?>
<QuickLicenseManager>
<error>Details about the error</error>
</QuickLicenseManager>
```

AddUser

Adds a new user. To use a proxy server, you must set the UseProxyServer, ProxyUser, ProxyDomain and ProxyPassword properties prior to calling this function.

C#: void AddUser(string webServiceUrl, string customerName, string customerEmail, string customerPhone, string customerFax, string customerMobile, string customerCompany, string customerAddress1, string customerAddress2, string customerCity, string customerState, string customerZip, string customerCountry, string customerIP, string customerNotes, bool includeInMailList, out string response)

Parameters

webServiceUrl - URL to the QLM web service

customerName - Full Name

customerEmail - Email address

customerPhone - Phone number

customerFax - Fax number

customerMobile - Mobile phone number

customerCompany - Company name

customerAddress1 - Address 1

customerAddress2 - Address 2

customerCity - City

customerState - State

customerZip - Zip Code

customerCountry - Country

customerIP - IP Address

customerNotes - Notes

includeInMailList - Include in mail list

webServiceUrl - URL to the QLM Web service.

activationKey - the license key to activate

computerID - the unique computer identifier

response - XML fragment containing the result of the call. The Xml fragment schema is as follows:

```
<?xml version='1.0' encoding='UTF-8'?>
<QuickLicenseManager>
<result>Customer ABC was added successfully.".</result>
</QuickLicenseManager>
```

In the event of an error, the XML fragments returns:

```
<?xml version='1.0' encoding='UTF-8'?>
<QuickLicenseManager>
<error>Details about the error</error>
</QuickLicenseManager>
```

GetLatestVersion

Gets the latest version of the specified product.

C#: string GetLatestVersion (string webServiceUrl, int productID, int majorVersion, int minorVersion, out string downloadUrl, out string notes, out string response)

Parameters

webServiceUrl - URL to the QLM Web service.

productID - ID of the product.

majorVersion- Major version of the product.

minorVersion - Minor version of the product.

downloadUrl - returns the url to download the latest version.

notes - returns notes about the latest version.

response - XML fragment containing the result of the call. The Xml fragment schema is as follows:

Returns the value of the latest version.

```
<?xml version='1.0' encoding='UTF-8'?>
<QuickLicenseManager>
<result>The latest version is xyz.</result>
<latestVersion>xyz</latestVersion>
</QuickLicenseManager>
```

In the event of an error, the XML fragments returns:

```
<?xml version='1.0' encoding='UTF-8'?>
<QuickLicenseManager>
<error>Details about the error</error>
</QuickLicenseManager>
```

GetOrder

Connects to the web service and gets the status of an order. To use a proxy server, you must set the UseProxyServer, ProxyUser, ProxyDomain and ProxyPassword properties prior to calling this function.

C#: DataRowCollection GetOrder (string webServiceUrl, string orderID, out string response)

Parameters

webServiceUrl - URL to the QLM Web service.

orderID - ID of the order

response - XML fragment containing the result of the call. The Xml fragment schema is as follows:

```
<?xml version='1.0' encoding='UTF-8'?>
<QuickLicenseManager>
<result>Order status updated successfully.</result>
</QuickLicenseManager>
```

In the event of an error, the XML fragments returns:

```
<?xml version='1.0' encoding='UTF-8'?>
<QuickLicenseManager>
<error>The order id is not valid</error>
</QuickLicenseManager>
```

Return

DataRowCollection containing all the data associated with this order.

Example:

```
QlmLicense license = new QlmLicense ();
license.DefineProduct(1, "Demo", 1, 0, "DemoKey", "{24EAA3C1-3DD7-40E0-AEA3-D20AA17A6005}");
string response = string.Empty;
DataRowCollection drc = license.GetOrder ("http://quicklicensemanager.com/qlmdemov6/qlmservice.asmx",
1234, out response);
if (drc != null)
{
  foreach (DataRow dr in drc)
  {
          string activationKey = dr["ActivationKey"];
  }
}
```

GetUserData

Gets the UserData1 field for a specific license key.

C#: string GetUserData (string webServiceUrl, string activationKey, out string response)

Parameters

webServiceUrl - URL to the QLM Web service.

activationKey - activation key to query.

response - XML fragment containing the result of the call. The Xml fragment schema is as follows:

Returns the value of the userData1 field.

```
<?xml version='1.0' encoding='UTF-8'?>
<QuickLicenseManager>
<result>Successfully executed query using filter...</result>
</QuickLicenseManager>
```

In the event of an error, the XML fragments returns:

```
<?xml version='1.0' encoding='UTF-8'?>
<QuickLicenseManager>
<error>Details about the error</error>
</QuickLicenseManager>
```

GetMaintenancePlanRenewalDate

Gets the maintenance plan renewal date.

C#: DateTime GetMaintenancePlanRenewalDate (string webServiceUrl, string activationKey)

Parameters

webServiceUrl - URL to the QLM Web service.

activationKey - the activation key of the record.

Note that the date returned is a UTC date. In the event the maintenance plan date is not set, the return value is set to DateTime.MinValue which is Jan 1 0001.

GetMaintenancePlanRenewalDateByComputerKey

Gets the maintenance plan renewal date given a computer key.

C#: DateTime GetMaintenancePlanRenewalDateByComputerKey (string webServiceUrl, string computerKey)

Parameters

webServiceUrl - URL to the QLM Web service.

computerKey - the computer key of the record.

Note that the date returned is a UTC date. In the event the maintenance plan date is not set, the return value is set to DateTime.MinValue which is Jan 1 0001.

GetUserDataFromActivationLog

Gets the UserData1 field for a specific license key from the ActivationLog table. The ActivationLog table is used when a license key is of type MultipleActivationsKey and more than 1 seat is associated to the license key.

C#: string GetUserDataFromActivationLog (string webServiceUrl, string activationKey, stromg computerID, out string response)

Parameters

webServiceUrl - URL to the QLM Web service.

activationKey - activation key to query.

computerID - Unique identifier of the computer on which the license was activated.

response - XML fragment containing the result of the call. The Xml fragment schema is as follows:

Returns the value of the userData1 field.

```
<?xml version='1.0' encoding='UTF-8'?>
<QuickLicenseManager>
<result>Successfully executed query using filter...</result>
</QuickLicenseManager>
```

In the event of an error, the XML fragments returns:

```
<?xml version='1.0' encoding='UTF-8'?>
<QuickLicenseManager>
<error>Details about the error</error>
</QuickLicenseManager>
```

IsIllegalComputer

Connects to the web service and checks if the current computer is properly registered in the QLM database. To use a proxy server, you must set the UseProxyServer, ProxyUser, ProxyDomain and ProxyPassword properties prior to calling this function.

C#: void IsIllegalComputer (string webServiceUrl, string activationKey, string computerKey, string computerID, string computerName, string qlmVersion, out string response)

Parameters

webServiceUrl - URL to the QLM Web service.

activationKey - the activation key

computerKey - the computer bound key

computerID - the computer identifier

computerName - the computer name

qlmVersion - the version of the QLM engine

response - XML fragment containing the result of the call. The Xml fragment schema is as follows:

```
<?xml version='1.0' encoding='UTF-8'?>
<QuickLicenseManager>
<result>The activation key is valid.</result>
</QuickLicenseManager>
```

In the event of an error, the XML fragments returns:

```
<?xml version='1.0' encoding='UTF-8'?>
<QuickLicenseManager>
<error>The activation key is not valid</error>
</QuickLicenseManager>
```

IsLicenseKeyValid

Connects to the web service and checks if the provided license key is valid. To use a proxy server, you must set the UseProxyServer, ProxyUser, ProxyDomain and ProxyPassword properties prior to calling this function.

C#: void IsLicenseKeyValid (string webServiceUrl, string activationKey, out string response)

Parameters

webServiceUrl - URL to the QLM Web service.

activationKey - the license key to activate

response - XML fragment containing the result of the call. The Xml fragment schema is as follows:

```
<?xml version='1.0' encoding='UTF-8'?>
<QuickLicenseManager>
<result>The activation key is valid.</result>
</QuickLicenseManager>
```

In the event of an error, the XML fragments returns:

```
<?xml version='1.0' encoding='UTF-8'?>
<QuickLicenseManager>
<error>The activation key is not valid</error>
</QuickLicenseManager>
```

ReleaseLicense

Releases a license key over the internet. To use a proxy server, you must set the UseProxyServer, ProxyUser, ProxyDomain and ProxyPassword properties prior to calling this function.

C#: void ReleaseLicense (string webServiceUrl, string activationKey, string computerID, out string response)

Parameters

webServiceUrl - URL to the QLM Web service.

activationKey - the license key to activate

computerID - the unique computer identifier

response - XML fragment containing the result of the call. The Xml fragment schema is as follows:

```
<?xml version='1.0' encoding='UTF-8'?>
<QuickLicenseManager>
<result>ActivationKey A162DCF05C30D371A2D0E0461040A0 has been released.</result>
</QuickLicenseManager>
```

In the event of an error, the XML fragments returns:

```
<?xml version='1.0' encoding='UTF-8'?>
<QuickLicenseManager>
<error>Details about the error</error>
</QuickLicenseManager>
```

SubscribeToMailList

Subscribes or unsubscribes a user from the mail list. To use a proxy server, you must set the UseProxyServer, ProxyUser, ProxyDomain and ProxyPassword properties prior to calling this function.

C#: bool SubscribeToMailList (string webServiceUrl, string customerEmail, bool includeInMailList, out string response)

Http: http://server/qlm/qlmservice.asmx/SubscribeToMailListHttp?is_email=user@cie.com&is_include=1

Parameters

webServiceUrl - URL to the QLM Web service.

customerEmail - email address of customer

includeInMailList - true to subscribe, false to unsubscribe

response - XML fragment containing the result of the call. The Xml fragment schema is as follows:

```
<?xml version='1.0' encoding='UTF-8'?>
<QuickLicenseManager>
<result>Succesfully subscribed customer.</result>
</QuickLicenseManager>
```

In the event of an error, the XML fragments returns:

```
<?xml version='1.0' encoding='UTF-8'?>
<QuickLicenseManager>
<error>Details about the error</error>
</QuickLicenseManager>
```

UpdateUser

Updates the data of an existing user. To use a proxy server, you must set the UseProxyServer, ProxyUser, ProxyDomain and ProxyPassword properties prior to calling this function.

C#: void UpdateUser(string webServiceUrl, string previousEmail, string customerName, string customerEmail, string customerPhone, string customerFax, string customerMobile, string customerCompany, string customerAddress1, string customerAddress2, string customerCity, string customerState, string customerZip, string customerCountry, string customerIP, string customerNotes, bool includeInMailList, out string response)

Parameters

webServiceUrl - URL to the QLM web service

previousEmail - Email address of the existing user to update

customerName - Full Name

customerEmail - Email address

customerPhone - Phone number

customerFax - Fax number

customerMobile - Mobile phone number

customerCompany - Company name

customerAddress1 - Address 1

customerAddress2 - Address 2

customerCity - City

customerState - State

customerZip - Zip Code

customerCountry - Country

customerIP - IP Address

customerNotes - Notes

includeInMailList - Include in mail list

response - XML fragment containing the result of the call. The Xml fragment schema is as follows:

```
<?xml version='1.0' encoding='UTF-8'?>
<QuickLicenseManager>
<result>Customer ABC was updated successfully.".</result>
</QuickLicenseManager>
```

In the event of an error, the XML fragments returns:

```
<?xml version='1.0' encoding='UTF-8'?>
<QuickLicenseManager>
<error>Details about the error</error>
</QuickLicenseManager>
```

Web Service Management API

The QLM Web Service provides an extended set of functions that you may want to use for managing your license keys or integrating licensing with any other internal process that you may have. This extended set of functions should typically not be part of your application but rather called from a server or a system that your end-user does not have access to.

Before calling any of the functions below, you need to set the adminEncryptionKey property of the QlmLicense object to the value specified in your Site Properties (Manage Keys tab / Site). If you are developing an application in .NET, it is highly recommeded that you obfuscate your code, and more specifically that you encrypt sensitive strings such as the adminEncryptionKey.

CreateActivationKey

Creates an activation key over the internet. To use a proxy server, you must set the UseProxyServer, ProxyUser, ProxyDomain and ProxyPassword properties prior to calling this function.

Note that to call this function, you must update the web.config on the web server as follows:

<setting name="enableCreateActivationKey" serializeAs="String">
<value>True</value>
</setting>

C#: void CreateActivationKey(string webServiceUrl, string email, int features, int quantity, bool useMultipleActivationsKey, string qlmVersion, string vendor, string userData1, string affiliateID, out string response)

Parameters

webServiceUrl - URL to the QLM Web service.

email - email address to associate to the license key - may be empty

features - or'ed value of the features to enable

quantity - the number of licenses to embed in the key

useMultipleActivationsKey - if set to true and quantity > 1, one license key will be generated for all required licenses. The number of licenses will be embedded in the license key

qlmVersion - the version of the QLM Engine

vendor - the eCommerce vendor to use when generating the key

userData1 - user data to associate to this license

affiliateID - ID of affiliate

response - XML fragment containing the result of the call. The Xml fragment schema is as follows:

```
<?xml version='1.0' encoding='UTF-8'?>
<QuickLicenseManager>
<keys>A062E-9D0CC-6DC80-0D6A0-E0701-000A0;A062E-9D0CC-6DC80-0D6A0-E0701-000A0</ke
ys>
</QuickLicenseManager>
```

In the event of an error, the XML fragments returns:

```
<?xml version='1.0' encoding='UTF-8'?>
<QuickLicenseManager>
<error>Details about the error</error>
</QuickLicenseManager>
```

CreateActivationKeyEx

Creates an activation key over the internet. To use a proxy server, you must set the UseProxyServer, ProxyUser, ProxyDomain and ProxyPassword properties prior to calling this function.

Note that to call this function, you must update the web.config on the web server as follows:

<setting name="enableCreateActivationKey" serializeAs="String">
<value>True</value>
</setting>

C#: void CreateActivationKeyEx(string webServiceUrl, string email, int [] features, int quantity, bool useMultipleActivationsKey, string qlmVersion, string vendor, string userData1, string affiliateID, out string response)

Parameters

webServiceUrl - URL to the QLM Web service.

email - email address to associate to the license key - may be empty

features - array of feature sets. each feature set is an or'ed value of the features to enable in the feature set

quantity - the number of licenses to embed in the key

useMultipleActivationsKey - if set to true and quantity > 1, one license key will be generated for all required licenses. The number of licenses will be embedded in the license key

qlmVersion - the version of the QLM Engine

vendor - the eCommerce vendor to use when generating the key

userData1 - user data to associate to this license

affiliateID - ID of affiliate

response - XML fragment containing the result of the call. The Xml fragment schema is as follows:

```
<?xml version='1.0' encoding='UTF-8'?>
<QuickLicenseManager>
<keys>A062E-9D0CC-6DC80-0D6A0-E0701-000A0;A062E-9D0CC-6DC80-0D6A0-E0701-000A0</keys>
</QuickLicenseManager>
```

In the event of an error, the XML fragments returns:

```
<?xml version='1.0' encoding='UTF-8'?>
<QuickLicenseManager>
<error>Details about the error</error>
</QuickLicenseManager>
```

CreateActivationKeyWithExpiryDate

Creates an activation key with an expiry date over the internet. To use a proxy server, you must set the UseProxyServer, ProxyUser, ProxyDomain and ProxyPassword properties prior to calling this function.

Note that to call this function, you must update the web.config on the web server as follows:

<setting name="enableCreateActivationKey" serializeAs="String">
<value>True</value>
</setting>

C#: void CreateActivationKeyWithExpiryDate(string webServiceUrl, string email, int features, int quantity, bool useMultipleActivationsKey, string qlmVersion, string vendor, string userData1, string affiliateID, DateTime expiryDate, int expiryDuration, out string response)

Parameters

webServiceUrl - URL to the QLM Web service.

email - email address to associate to the license key - may be empty

features - or'ed value of the features to enable

quantity - the number of licenses to embed in the key

useMultipleActivationsKey - if set to true and quantity > 1, one license key will be generated for all required licenses. The number of licenses will be embedded in the license key

qlmVersion - the version of the QLM Engine

vendor - the eCommerce vendor to use when generating the key

userData1 - user data to associate to this license

affiliateID - ID of affiliate

expiryDate - Expiry date of the key

expiryDuration - Expiry duration of the key

response - XML fragment containing the result of the call. The Xml fragment schema is as follows:

```
<?xml version='1.0' encoding='UTF-8'?>
<QuickLicenseManager>
<keys>A062E-9D0CC-6DC80-0D6A0-E0701-000A0;A062E-9D0CC-6DC80-0D6A0-E0701-000A0</keys>
</QuickLicenseManager>
```

In the event of an error, the XML fragments returns:

```
<?xml version='1.0' encoding='UTF-8'?>
<QuickLicenseManager>
<error>Details about the error</error>
</QuickLicenseManager>
```

CreateActivationKeyWithExpiryDateEx

Creates an activation key with an expiry date over the internet. To use a proxy server, you must set the UseProxyServer, ProxyUser, ProxyDomain and ProxyPassword properties prior to calling this function.

Note that to call this function, you must update the web.config on the web server as follows:

<setting name="enableCreateActivationKey" serializeAs="String">
<value>True</value>
</setting>

C#: void CreateActivationKeyWithExpiryDateEx(string webServiceUrl, string email, int [] features, int quantity, bool useMultipleActivationsKey, string qlmVersion, string vendor, string userData1, string affiliateID, DateTime expiryDate, int expiryDuration, out string response)

Parameters

webServiceUrl - URL to the QLM Web service.

email - email address to associate to the license key - may be empty

features - array of feature sets. each feature set is an or'ed value of the features to enable in the feature set

quantity - the number of licenses to embed in the key

useMultipleActivationsKey - if set to true and quantity > 1, one license key will be generated for all required licenses. The number of licenses will be embedded in the license key

qlmVersion - the version of the QLM Engine

vendor - the eCommerce vendor to use when generating the key

userData1 - user data to associate to this license

affiliateID - ID of affiliate

expiryDate - Expiry date of the key

expiryDuration - Expiry duration of the key

response - XML fragment containing the result of the call. The Xml fragment schema is as follows:

```
<?xml version='1.0' encoding='UTF-8'?>
<QuickLicenseManager>
<keys>A062E-9D0CC-6DC80-0D6A0-E0701-000A0;A062E-9D0CC-6DC80-0D6A0-E0701-000A0</keys>
</QuickLicenseManager>
```

In the event of an error, the XML fragments returns:

```
<?xml version='1.0' encoding='UTF-8'?>
<QuickLicenseManager>
<error>Details about the error</error>
</QuickLicenseManager>
```

CreateOrder

Creates an order and an activation key with an expiry date over the internet. To use a proxy server, you must set the UseProxyServer, ProxyUser, ProxyDomain and ProxyPassword properties prior to calling this function.

Note that to call this function, you must update the web.config on the web server as follows:

```
<setting name="enableCreateOrder" serializeAs="String">
<value>True</value>
</setting>
```

C#: void CreateOrder(string webServiceUrl, string email, int features, int quantity, bool useMultipleActivationsKey, string qlmVersion, string vendor, string userData1, string affiliateID, DateTime expiryDate, int expiryDuration, string orderID, int orderStatus, out string response)

Parameters

webServiceUrl - URL to the QLM Web service.

email - email address to associate to the license key - may be empty

features - or'ed value of the features to enable

quantity - the number of licenses to embed in the key

useMultipleActivationsKey - if set to true and quantity > 1, one license key will be generated for all required licenses. The number of licenses will be embedded in the license key

qlmVersion - the version of the QLM Engine

vendor - the eCommerce vendor to use when generating the key

userData1 - user data to associate to this license

affiliateID - ID of affiliate

expiryDate - Expiry date of the key

expiryDuration - Expiry duration of the key

orderID - order ID

orderStatus - status of the order. Possible values are: None (0), In Progress (1), Completed (2).

response - XML fragment containing the result of the call. The Xml fragment schema is as follows:

```
<?xml version='1.0' encoding='UTF-8'?>
<QuickLicenseManager>
<keys>A062E-9D0CC-6DC80-0D6A0-E0701-000A0;A062E-9D0CC-6DC80-0D6A0-E0701-000A0</ke
ys>
</QuickLicenseManager>
```

In the event of an error, the XML fragments returns:

```
<?xml version='1.0' encoding='UTF-8'?>
<QuickLicenseManager>
<error>Details about the error</error>
</QuickLicenseManager>
```

CreateOrderEx

Creates an order and an activation key with an expiry date over the internet. To use a proxy server, you must set the UseProxyServer, ProxyUser, ProxyDomain and ProxyPassword properties prior to calling this function.

Note that to call this function, you must update the web.config on the web server as follows:

<setting name="enableCreateOrder" serializeAs="String">
<value>True</value>
</setting>

C#: void CreateOrderEx(string webServiceUrl, string email, int [] features, int quantity, bool useMultipleActivationsKey, string qlmVersion, string vendor, string userData1, string affiliateID, DateTime expiryDate, int expiryDuration, string orderID, int orderStatus, out string response)

Parameters

webServiceUrl - URL to the QLM Web service.

email - email address to associate to the license key - may be empty

features - array of feature sets. each feature set is an or'ed value of the features to enable in the feature set

quantity - the number of licenses to embed in the key

useMultipleActivationsKey - if set to true and quantity > 1, one license key will be generated for all required licenses. The number of licenses will be embedded in the license key

qlmVersion - the version of the QLM Engine

vendor - the eCommerce vendor to use when generating the key

userData1 - user data to associate to this license

affiliateID - ID of affiliate

expiryDate - Expiry date of the key

expiryDuration - Expiry duration of the key

orderID - order ID

orderStatus - status of the order. Possible values are: None (0), In Progress (1), Completed (2).

response - XML fragment containing the result of the call. The Xml fragment schema is as follows:

```
<?xml version='1.0' encoding='UTF-8'?>
<QuickLicenseManager>
<keys>A062E-9D0CC-6DC80-0D6A0-E0701-000A0;A062E-9D0CC-6DC80-0D6A0-E0701-000A0</ke
ys>
</QuickLicenseManager>
```

In the event of an error, the XML fragments returns:

```
<?xml version='1.0' encoding='UTF-8'?>
<QuickLicenseManager>
<error>Details about the error</error>
</QuickLicenseManager>
```

DeleteLicense

Deletes a license key over the internet. To use a proxy server, you must set the UseProxyServer, ProxyUser, ProxyDomain and ProxyPassword properties prior to calling this function.

C#: void DeleteLicense (string webServiceUrl, string activationKey, string computerID, bool multipleActivationsKey, bool historyTable, out string response)

Parameters

webServiceUrl - URL to the QLM Web service.

activationKey - the license key to delete

computerID - the computer ID to delete. The computerID is optional if multipleActivationsKey is false.

multipleActivationsKey - set to true if this key is a multiple activations key. The Delete operates then on the ActivationLog table. The computerID must be specified.

historyTable - set to true if you want to delete licenses from the history table where all released licenses are backed up.

response - XML fragment containing the result of the call. The Xml fragment schema is as follows:

```
<?xml version='1.0' encoding='UTF-8'?>
<QuickLicenseManager>
<result>ActivationKey A162DCF05C30D371A2D0E0461040A0 has been deleted.</result>
</QuickLicenseManager>
```

In the event of an error, the XML fragments returns:

```
<?xml version='1.0' encoding='UTF-8'?>
<QuickLicenseManager>
<error>Details about the error</error>
</QuickLicenseManager>
```

DownloadProducts

Downloads the list of products from the web service over the internet. To use a proxy server, you must set the UseProxyServer, ProxyUser, ProxyDomain and ProxyPassword properties prior to calling this function.

C#: void DownloadProducts (string webServiceUrl, ref string dataSet, out string response)

Parameters

webServiceUrl - URL to the QLM Web service.

serverDate - returns the date the products were uploaded to the server.

dataSet - returned dataset containing products

response - XML fragment containing the result of the call. The Xml fragment schema is as follows:

```
<?xml version='1.0' encoding='UTF-8'?>
<QuickLicenseManager>
<result>Succesfully downloaded products.</result>
</QuickLicenseManager>
```

In the event of an error, the XML fragments returns:

```
<?xml version='1.0' encoding='UTF-8'?>
<QuickLicenseManager>
<error>Details about the error</error>
</QuickLicenseManager>
```

GetCustomersInfo

Retrieves information about a set of customers.

C#: DataSet GetCustomersInfo (string webServiceUrl, string fieldName, string fieldOperator, string fieldValue, out string response)

Parameters

webServiceUrl - URL to the QLM Web service.

field - field from the Customers table. Typical fields are: Email, FullName, CustomerID

fieldOperator - a valid SQL operator such as: =, like, <>

fieldValue - value of the field to match

response - XML fragment containing the result of the call. The Xml fragment schema is as follows:

```
<?xml version='1.0' encoding='UTF-8'?>
<QuickLicenseManager>
<result>Successfully executed query ...</result>
</QuickLicenseManager>
```

In the event of an error, the XML fragments returns:

```
<?xml version='1.0' encoding='UTF-8'?>
<QuickLicenseManager>
<error>Details about the error</error>
</QuickLicenseManager>
```

Example

```
DataSet ds = license.GetCustomersInfo (webServiceUrl, "email", "=", "'customer@mail.com'", out
response);
```

GetDataSet

Gets a data set in xml format that meets the criteria specified in the filter.

C#: void GetDataSet (string webServiceUrl, string filter, ref string dataSet, out string response)

Parameters

webServiceUrl - URL to the QLM Web service.

filter - SQL filter to determine which records to return. Use a where clause sql syntax, example: ActivationKey='AAAA'

dataSet - returned dataset containing license key records that match criteria

response - XML fragment containing the result of the call. The Xml fragment schema is as follows:

```
<?xml version='1.0' encoding='UTF-8'?>
<QuickLicenseManager>
<result>Successfully executed query using filter...</result>
</QuickLicenseManager>
```

In the event of an error, the XML fragments returns:

```
<?xml version='1.0' encoding='UTF-8'?>
<QuickLicenseManager>
<error>Details about the error</error>
</QuickLicenseManager>
```

GetDataSetEx

Gets a data set in xml format that meets the criteria specified in the filter.

C#: void GetDataSetEx (string webServiceUrl, string table, string filter, ref string dataSet, out string response)

Parameters

webServiceUrl - URL to the QLM Web service.

table- specify the table to query. The value can be: LicenseKeys or LicenseKeysHistory. LicenseKeys is the table where all license keys are recorded. LicenseKeysHistory contains all the released license keys.

filter - SQL filter to determine which records to return. Use a where clause sql syntax, example: ActivationKey='AAAA'

dataSet - returned dataset containing license key records that match criteria

response - XML fragment containing the result of the call. The Xml fragment schema is as follows:

```
<?xml version='1.0' encoding='UTF-8'?>
<QuickLicenseManager>
<result>Successfully executed query using filter...</result>
</QuickLicenseManager>
```

In the event of an error, the XML fragments returns:

```
<?xml version='1.0' encoding='UTF-8'?>
<QuickLicenseManager>
<error>Details about the error</error>
</QuickLicenseManager>
```

RenewSubscription

Connects to the web service and renews a subscription.

When a subscription is renewed, each activated license is automatically reactivated on the server and a new computer bound key is generated with a new expiry date. When customers reactivate their license, they receive the new computer bound key with the new expiry date, thus extending their subscription period.

C#: bool RenewSubscription (string webServiceUrl, string activationKey, DateTime expiryDate, out string errorMessage)

webServiceUrl - URL to the QLM Web service.

activationKey- activation key to extend

expiryDate - Expiry date of the subscription

errorMessage - Error message if the operation failed.

Return

True if the subscription renewal was successful.

SetMaintenancePlanRenewalDate

Sets the maintenance plan renewal date.

C#: bool SetMaintenancePlanRenewalDate (string webServiceUrl, string activationKey, DateTime maintenancePlanRenewalDate, out string errorMessage)

Parameters

webServiceUrl - URL to the QLM Web service.

activationKey - the activation key of the record.

maintenancePlanRenewalDate - the renewal date of the maintenance plan

errorMessage - returned error message

It is recommended to send a UTC date.

If the function succeeds, the return value is true.If the function fails, the return is value is false. The errorMessage contains details about the error.

UpdateActivationLogInfo

Updates the data associated with a license key. To use a proxy server, you must set the UseProxyServer, ProxyUser, ProxyDomain and ProxyPassword properties prior to calling this function.

The ActivationLog table is used when multiple licenses are issued from a single ActivationKey. In this case, the data associated with each activated computer is stored in the ActivationLog table instead of the LicenseKeys table. Therefore, to update data in the ActivationLog table, you need to specify which computer to update. The computerID, computerKey and computerName arguments can be specified to identify the computer. At least one of these arguments must be specified.

The ActivationLog table contains the following updatable fields:

ComputerKey, ComputerName, ComputerID, ActivationDate, LastAccessedDate, ActivationCount

C#: bool UpdateActivationLogInfo (string webServiceUrl, string activationKey, string computerID, string computerKey, string computerName, string licenseData, out string response)

Parameters

webServiceUrl - URL to the QLM Web service.

activationKey- the license key to udpate

computerID- the ID the computer to udpate

computerKey- the computer bound key to udpate

computerName- the computer name to udpate

licenseData - XML fragment containing the fields to update. The XML fragment should be of the form:

```
<licenseArguments
    field1=" 'value'
    field2=" 'value' "
</licenseArguments>
```

where field1 is the name of a field in the LicenseKeys table. For fields of type date, you should use the following date/time format: yyyy-MM-dd HH:mm:ss

Example:

```
<licenseArguments
    ComputerName= " 'my pc' "
    UserData1=" 'my user data' "
</licenseArguments>
```

response - XML fragment containing the result of the call. The Xml fragment schema is as follows:

```
<?xml version='1.0' encoding='UTF-8'?>
<QuickLicenseManager>
<result>Successfully updated license information for ActivationKey=XYZ.</result>
</QuickLicenseManager>
```

In the event of an error, the XML fragments returns:

```
<?xml version='1.0' encoding='UTF-8'?>
<QuickLicenseManager>
<error>Details about the error</error>
</QuickLicenseManager>
```

UpdateLicenseInfo

Updates the data associated with a license key. To use a proxy server, you must set the UseProxyServer, ProxyUser, ProxyDomain and ProxyPassword properties prior to calling this function.

C#: bool UpdateLicenseInfo (string webServiceUrl, string activationKey, string licenseData, out string response)

Parameters

webServiceUrl - URL to the QLM Web service.

activationKey- the license key to udpate

licenseData - XML fragment containing the fields to update. The XML fragment should be of the form:

```
<licenseArguments

    field1=" 'value'

    field2=" 'value' "

</licenseArguments>
```

where field1 is the name of a field in the LicenseKeys table. For fields of type date, you should use the following date/time format: yyyy-MM-dd HH:mm:ss

Example:

```
<licenseArguments

    Comment= " 'my comment' "

    UserData1=" 'my user data' "

    OrderDate=" '2008-3-12 21:14:58' "

</licenseArguments>
```

response - XML fragment containing the result of the call. The Xml fragment schema is as follows:

```
<?xml version='1.0' encoding='UTF-8'?>
<QuickLicenseManager>
<result>Successfully updated license information for ActivationKey=XYZ.</result>
</QuickLicenseManager>
```

In the event of an error, the XML fragments returns:

```
<?xml version='1.0' encoding='UTF-8'?>
<QuickLicenseManager>
<error>Details about the error</error>
</QuickLicenseManager>
```

UpdateLicenseKey

Updates a license key with another license key. To use a proxy server, you must set the UseProxyServer, ProxyUser, ProxyDomain and ProxyPassword properties prior to calling this function.

C#: bool UpdateLicenseKey (string webServiceUrl, string currentKey, string newKey, out string response)

Parameters

webServiceUrl - URL to the QLM Web service.

currentKey - the current license key

newKey - the new license key

response - XML fragment containing the result of the call. The Xml fragment schema is as follows:

```
<?xml version='1.0' encoding='UTF-8'?>
<QuickLicenseManager>
<result>Successfully updated license information for ActivationKey=XYZ.</result>
</QuickLicenseManager>
```

In the event of an error, the XML fragments returns:

```
<?xml version='1.0' encoding='UTF-8'?>
<QuickLicenseManager>
<error>Details about the error</error>
</QuickLicenseManager>
```

UpdateOrderStatus

Connects to the web service and updates the status of an order. To use a proxy server, you must set the UseProxyServer, ProxyUser, ProxyDomain and ProxyPassword properties prior to calling this function.

C#: bool UpdateOrderStatus (string webServiceUrl, string orderID, int orderStatus, out string response)

Parameters

webServiceUrl - URL to the QLM Web service.

orderID - ID of the order

orderStatus - status of the order to set

response - XML fragment containing the result of the call. The Xml fragment schema is as follows:

```
<?xml version='1.0' encoding='UTF-8'?>
<QuickLicenseManager>
<result>Order status updated successfully.</result>
</QuickLicenseManager>
```

In the event of an error, the XML fragments returns:

```
<?xml version='1.0' encoding='UTF-8'?>
<QuickLicenseManager>
<error>The order id is not valid</error>
</QuickLicenseManager>
```

Return

Boolean indicating whether the operation was successful.

Example:

```
QlmLicense license = new QlmLicense ();
license.DefineProduct(1, "Demo", 1, 0, "DemoKey", "{24EAA3C1-3DD7-40E0-AEA3-D20AA17A6005}");
string response = string.Empty;
bool stat  = license.UpdateOrderStatus ("http://quicklicensemanager.com/qlmdemov6/qlmservice.asmx",
1234, 2, out response);
```

UpgradeLicense

Connects to the web service and upgrades a license. You can upgrade the following data associated to a license:

- Features associated to a license
- Expiry date of the license
- Duration of the license
- Major and Minor version of the product
- The version of the QLM Engine used to generate the license key

When a license is upgraded, a new license key is generated and replaces the existing one. The old license is copied to the released licenses table.

To use a proxy server, you must set the UseProxyServer, ProxyUser, ProxyDomain and ProxyPassword properties prior to calling this function.

C#: bool UpgradeLicense (string webServiceUrl, string activationKey, int productID, int majorVersion, int minorVersion, string qlmVersion, int[] features, DateTime dtExpiry, int expiryDuration, out string response)

Http:
hhttp://server/qlm/qlmservice.asmx/UpgradeLicense?is_avkey=<activationKey>&is_productid=<pid>&is_majorversion=<majorVersion>&is_minorversion=<minorVersion>&is_expdate=<yyyy-mm-dd>&is_expdate>&is_expduration=<expiry duration>

Example:
http://server/qlm/qlmservice.asmxUpgradeLicense?is_avkey=B0739A30F960FAA0FA3045D0560000000&is_productid=1&is_majorversion=1&is_minorversion=0&is_expdate=2008-06-01

Parameters

webServiceUrl - URL to the QLM Web service.

activationKey- activation key to update

productID - ID of the product

majorVersion - New major version of the product

minorVersion - New minor version of the product

qlmVersion - Version of the QLM Engine to use.

features - An array of feature sets specifying the features that should be enabled in the created key. Each feature has a unique feature set and ID associated to it. To combine features, perform a bitwise OR operation on the required features.

dtExpiry - Expiry date of the license key

expiryDuration - Expiry duration of the license key

response - XML fragment containing the result of the call. The Xml fragment schema is as follows:

```
<?xml version='1.0' encoding='UTF-8'?>
<QuickLicenseManager>
<result>The license was upgraded...</result>
</QuickLicenseManager>
```

In the event of an error, the XML fragments returns:

```
<?xml version='1.0' encoding='UTF-8'?>
<QuickLicenseManager>
<error>The activation key is not valid</error>
</QuickLicenseManager>
```

Return

True if the upgrade was successful.

UpgradeLicenseEx

Connects to the web service and upgrades a license. You can upgrade the following data associated to a license:

- Features associated to a license
- Expiry date of the license
- Duration of the license
- Major and Minor version of the product
- The version of the QLM Engine used to generate the license key

When a license is upgraded, a new license key is generated and replaces the existing one. The old license is copied to the released licenses table.

To use a proxy server, you must set the UseProxyServer, ProxyUser, ProxyDomain and ProxyPassword properties prior to calling this function.

C#: bool UpgradeLicense (string webServiceUrl, string activationKey, int productID, int majorVersion, int minorVersion, string qlmVersion, string features, DateTime dtExpiry, int expiryDuration, out string response)

Http:
hhttp://server/qlm/qlmservice.asmx/UpgradeLicense?is_avkey=<activationKey>&is_productid=<pid>&is_majorversion=<majorVersion>&is_minorversion=<minorVersion>&is_expdate=<yyyy-mm-dd>&is_expdate>&is_expduration=<expiry duration>

Example:
http://server/qlm/qlmservice.asmxUpgradeLicense?is_avkey=B0739A30F960FAA0FA3045D0560000000&is_productid=1&is_majorversion=1&is_minorversion=0&is_expdate=2008-06-01

Parameters

webServiceUrl - URL to the QLM Web service.

activationKey- activation key to update

productID - ID of the product

majorVersion - New major version of the product

minorVersion - New minor version of the product

qlmVersion - Version of the QLM Engine to use.

features - A semi comma separated list of enabled feature sets. For example, to enable features 001 and 002 in feature set 0 and features 004 and 008 in feature set 2, you use: "0:3;2:12"

dtExpiry - Expiry date of the license key

expiryDuration - Expiry duration of the license key

response - XML fragment containing the result of the call. The Xml fragment schema is as follows:

```
<?xml version='1.0' encoding='UTF-8'?>
<QuickLicenseManager>
<result>The license was upgraded...</result>
</QuickLicenseManager>
```

In the event of an error, the XML fragments returns:

```
<?xml version='1.0' encoding='UTF-8'?>
<QuickLicenseManager>
<error>The activation key is not valid</error>
</QuickLicenseManager>
```

Return

True if the upgrade was successful.

UploadProducts

Uploads the list of products to the web service over the internet. To use a proxy server, you must set the UseProxyServer, ProxyUser, ProxyDomain and ProxyPassword properties prior to calling this function.

C#: void UploadProducts (string webServiceUrl, string productsXml, DateTime updatedUtcDate, out string response)

Parameters

webServiceUrl - URL to the QLM Web service.

productsXml - >Xml file containing the list of products. The format of the XML file is identical to the products.xml file stored on the client side. For example:

```
<?xml version='1.0' encoding='UTF-8'?>
<LICENSES>
<PRODUCTS>
<PRODUCT Name="Demo" ID="1" Major="2" Minor="0" Key="DemoKey"
GUID="{AB932603-7336-4DA4-90C1-843C4146E388}" ReleaseDate="2008-12-11" Features=""
LatestVersion="2.0" LatestVersionUrl="" LatestVersionNotes="" />
<PRODUCT Name="Demo" ID="1" Major="1" Minor="0" Key="DemoKey"
GUID="{24EAA3C1-3DD7-40E0-AEA3-D20AA17A6005}" ReleaseDate="2007-12-01"
Features="0:1:F1;0:2:F2;0:4:F3;3:1:D1;" LatestVersion="1.1"
LatestVersionUrl="http://yourserver/setup.exe" LatestVersionNotes="In this field, insert comments that
describe the latest version of your product. This information can be retrieved by your application when
checking for updates and displayed to the end user." />
</PRODUCTS>
</LICENSES>
```

updatedUtcDate - UTC Date at which the products were last updated.

response - XML fragment containing the result of the call. The Xml fragment schema is as follows:

```
<?xml version='1.0' encoding='UTF-8'?>
<QuickLicenseManager>
<result>Succesfully downloaded products.</result>
</QuickLicenseManager>
```

In the event of an error, the XML fragments returns:

```
<?xml version='1.0' encoding='UTF-8'?>
<QuickLicenseManager>
<error>Details about the error</error>
</QuickLicenseManager>
```

Client Side API

QLM Pro provides a set of client side functions that you can use in your application or in any other order processing application that you may have. Client side functions do not communicate with the QLM Web Service.

Before calling any of the functions below, you need to call the DefineProduct function and set the PublicKey property.

Functions that create license keys such as CreateLicenseKey should not typically be used from within your application. They should be used from a server or a system that the end user does not have access to. In order to call any of the functions that create license keys, you must set the PrivateKey property of the QlmLicense object.

The Public and Private keys for your product can be found in the Define Products tab / Keys tab.

If you are developing an application in .NET, it is highly recommeded that you obfuscate your code, and more specifically that you encrypt sensitive strings such as the PrivateKey.

BackwardCompatible

Set this property to true to allow validation of keys prior to the latest version of the QLM engine.

C++: VARIANT_BOOL BackwardCompatible

C#: bool BackwardCompatible

CreateLicenseKey

Creates a non-computer bound license key. If the ExpiryDate is NULL and the ExpiryDuration is -1, the license key is a permanent non-evaluation license key.

Prior to calling this function, you must call DefineProduct and set the PrivateKey property. Note that including the PrivateKey in your code is not recommended. Creation of license keys should not typically be done from within the application but rather from a server that the user does not have access to.

C++: _bstr_t CreateLicenseKey (DATE ExpiryDate, int ExpiryDuration)

C#: string CreateLicenseKey (System.DateTime ExpiryDate, int ExpiryDuration)

Parameters

ExpiryDate - The date when the license will expire. Use NULL if you do not want to specify an expiry date.

ExpiryDuration - The duration of the evaluation period in days. Use -1 if you do not want to specify a duration.

Return

A non-computer bound license key.

CreateLicenseKeyEx

Creates a computer bound license key.

Prior to calling this function, you must call DefineProduct and set the PrivateKey property. Note that including the PrivateKey in your code is not recommended. Creation of license keys should not typically be done from within the application but rather from a server that the user does not have access to.

C++: _bstr_t CreateLicenseKeyEx (ELicenseType LicenseType, BSTR MachineID)

C#: string CreateLicenseKeyEx (ELicenseType LicenseType, string MachineID)

Parameters

LicenseType - Specify the type of license to generate. See the definition of LicenseType below.

MachineID - A unique identifier for the machine. If you specify a *ComputerName* as the LicenseType, this argument must be the Computer Name. If you specify *User Defined* as the LicenseType, this argument can be anything you want. When validating the license key in your code, you will need to provide the same value to the ValidateLicenseEx function.

Return

A computer bound license key.

CreateLicenseKeyEx2

Creates a computer bound license key that has an expiry date and a number of licenses.

Prior to calling this function, you must call DefineProduct and set the PrivateKey property. Note that including the PrivateKey in your code is not recommended. Creation of license keys should not typically be done from within the application but rather from a server that the user does not have access to.

C++: _bstr_t CreateLicenseKeyEx2 (DATE ExpiryDate, int ExpiryDuration, int NumberOfLicenses, ELicenseType LicenseType, BSTR MachineID)

C#: string CreateLicenseKeyEx2 (System.DateTime ExpiryDate, int ExpiryDuration, int NumberOfLicenses, ELicenseType LicenseType, string MachineID)

Parameters

ExpiryDate - The date when the license will expire. Use NULL if you do not want to specify an expiry date.

ExpiryDuration - The duration of the evaluation period in days. Use -1 if you do not want to specify a duration.

NumberOfLicenses - The number of licenses associated with the key. Use 1 if you do not want to use single activation licensing.

LicenseType - Specify the type of license to generate. See the definition of LicenseType below.

MachineID - A unique identifier for the machine. If you specify a *ComputerName* as the LicenseType, this argument must be the Computer Name. If you specify *User Defined* as the LicenseType, this argument can be anything you want. When validating the license key in your code, you will need to provide the same value to the ValidateLicenseEx function. To create a key that is not computer bound, set this argument to NULL and set the LicenseType to Generic.

Return

A computer bound license key.

CreateLicenseKeyEx3

Creates a computer bound license key that has an expiry date, a number of licenses and a specific set of features that are enabled.

Prior to calling this function, you must call DefineProduct and set the PrivateKey property. Note that including the PrivateKey in your code is not recommended. Creation of license keys should not typically be done from within the application but rather from a server that the user does not have access to.

C++: _bstr_t CreateLicenseKeyEx3 (DATE expiryDate, int expiryDuration, int numberOfLicenses, ELicenseType licenseType, BSTR machineID, in features)

C#: string CreateLicenseKeyEx3 (System.DateTime expiryDate, int expiryDuration, int numberOfLicenses, ELicenseType licenseType, string machineID, int features)

Parameters

ExpiryDate - The date when the license will expire. Use NULL if you do not want to specify an expiry date.

ExpiryDuration - The duration of the evaluation period in days. Use -1 if you do not want to specify a duration.

NumberOfLicenses - The number of licenses associated with the key. Use 1 if you do not want to use single activation licensing.

LicenseType - Specify the type of license to generate. See the definition of LicenseType below.

MachineID - A unique identifier for the machine. If you specify a *ComputerName* as the LicenseType, this argument must be the Computer Name. If you specify *User Defined* as the LicenseType, this argument can be anything you want. When validating the license key in your code, you will need to provide the same value to the ValidateLicenseEx function. To create a key that is not computer bound, set this argument to NULL and set the LicenseType to Generic.

Features - A value specifying the features that should be enabled in the created key. Each feature has a unique ID associated to it. To combine features, perform a bitwise OR operation on the required features.

Return

A computer bound license key.

CreateLicenseKeyEx4

Creates a computer bound license key that has an expiry date, a number of licenses and a specific set of features that are enabled.

Prior to calling this function, you must call DefineProduct and set the PrivateKey property. Note that including the PrivateKey in your code is not recommended. Creation of license keys should not typically be done from within the application but rather from a server that the user does not have access to.

C++: _bstr_t CreateLicenseKeyEx4 (DATE expiryDate, int expiryDuration, int numberOfLicenses, ELicenseType licenseType, BSTR machineID, int[] features)

C#: string CreateLicenseKeyEx4 (System.DateTime expiryDate, int expiryDuration, int numberOfLicenses, ELicenseType licenseType, string machineID, SAFEARRAY *Features)

Parameters

ExpiryDate - The date when the license will expire. Use NULL if you do not want to specify an expiry date.

ExpiryDuration - The duration of the evaluation period in days. Use -1 if you do not want to specify a duration.

NumberOfLicenses - The number of licenses associated with the key. Use 1 if you do not want to use single activation licensing.

LicenseType - Specify the type of license to generate. See the definition of LicenseType below.

MachineID - A unique identifier for the machine. If you specify a *ComputerName* as the LicenseType, this argument must be the Computer Name. If you specify *User Defined* as the LicenseType, this argument can be anything you want. When validating the license key in your code, you will need to provide the same value to the ValidateLicenseEx function. To create a key that is not computer bound, set this argument to NULL and set the LicenseType to Generic.

Features - An array of feature sets. Each feature set is a value specifying the features that should be enabled in the created key. The value of the feature set is the or'ed value of all the features to be enabled in the set. To combine features, perform a bitwise OR operation on the required features.

Return

A computer bound license key.

CreateLicenseKeyEx5

Creates a computer bound license key that has an expiry date, a number of licenses and a specific set of features that are enabled. This API is functionally identical to CreateLicenseKeyEx4. It was created to accomodate programming languages such as VB6 that cannot handle the array data type used in CreateLicenseKeyEx4.

Prior to calling this function, you must call DefineProduct and set the PrivateKey property. Note that including the PrivateKey in your code is not recommended. Creation of license keys should not typically be done from within the application but rather from a server that the user does not have access to.

C++: _bstr_t CreateLicenseKeyEx5 (DATE expiryDate, int expiryDuration, int numberOfLicenses, ELicenseType licenseType, BSTR machineID, BSTR features)

C#: string CreateLicenseKeyEx5 (System.DateTime expiryDate, int expiryDuration, int numberOfLicenses, ELicenseType licenseType, string machineID, string Features)

Parameters

ExpiryDate - The date when the license will expire. Use NULL if you do not want to specify an expiry date.

ExpiryDuration - The duration of the evaluation period in days. Use -1 if you do not want to specify a duration.

NumberOfLicenses - The number of licenses associated with the key. Use 1 if you do not want to use single activation licensing.

LicenseType - Specify the type of license to generate. See the definition of LicenseType below.

MachineID - A unique identifier for the machine. If you specify a *ComputerName* as the LicenseType, this argument must be the Computer Name. If you specify *User Defined* as the LicenseType, this argument can be anything you want. When validating the license key in your code, you will need to provide the same value to the ValidateLicenseEx function. To create a key that is not computer bound, set this argument to NULL and set the LicenseType to Generic.

Features - A set of features to be enabled using the following syntax:

<featureSet>:<featureValue>;<featureSet>:<featureValue>
Example: "0:8;1:2;3:14" - Enables: feature 8 in feature set 0, feature 2 in feature set 1 and features 2, 4, 8 in feature set 3.
To combine features, perform a bitwise OR operation on the required features.

Return

A computer bound license key.

DaysLeft

Returns the number of days left before the evaluation expires. You must call ValidateLicense prior to calling this function.

C++: int DaysLeft

C#: int DaysLeft ()

Return

Number of days left before the evaluation expires.

DefineProduct

The DefineProduct method initializes basic information required to validate license keys. You must call this function prior to calling any other function.

C++: HRESULT DefineProduct (int ProductID, BSTR ProductName, int MajorVersion, int MinorVersion, BSTR EncryptionKey, BSTR PersistenceKey)

C#: DefineProduct (int ProductID, string ProductName, int MajorVersion, int MinorVersion, string EncryptionKey, string PersistenceKey)

Parameters

ProductID: ID of the product as generated by Quick License Manager

ProductName: Name of the product

MajorVersion: Major version of the product (maximum 2 digits)

MinorVersion: Minor version of the product (maximum 2 digits)

Encryption Key: string used to encrypt the license key.

PersistenceKey: GUID associated with the product and automatically generated by Quick License Manager for each product. The evaluation information of the product is stored at runtime in the registry under HKCR\CLSID\<GUID>.

DeleteKeys

Deletes the license keys stored on the end user system with the StoreKeys API. To store keys, use the StoreKeys API. To read the stored keys, use the ReadKeys API.

C#: void DeleteKeys ()

Duration

Returns the duration in days of the evaluation key. You must call ValidateLicense prior to calling this function.

C++: int Duration

C#: int Duration ()

Return

Duration of the evaluation key.

ELicenseStatus

Enum of all possible values of the license key status. Note that the status can consist of a combination of these values:

EKeyPermanent : The license key is valid and it is a permanent license key.

EKeyInvalid : The license key is invalid. It was not decoded succesfully.

EKeyDemo : The license key is an evaluation key.

EKeyProductInvalid : The product ID of the license key does not correspond to the expected Product ID.

EKeyVersionInvalid : The Major or Minor version of the license key does not correspond to the expected Major or Minor version.

EKeyExpired : The license key has expired.

EKeyTampered: The license key was tampered typically indicating that the user is attempting to set the date back to run the software.

EKeyMachineInvalid: If you are using computer bound license keys, an EKeyMachineInvalid status indicates that the license key that was validated does not match the computer to which the license key was bound.

EKeyNeedsActivation: This flag indicates that the license key is an activation key. If you detect an activation key, you should not enable your application. You should just allow the user to activate his license. Once the license is activated, a computer bound key is issued. Once you detect a valid computer bound key, you can enable your application.

ELicenseType

Enum of all possible types of license keys.

Activation : The license key is a key that requires activation.

Evaluation (obsolete) : The license key is an evaluation key.

ComputerName : The license key is bound to the name of the computer.

Generic (previously PermanentGeneric) : The license key is permanent and not bound to a computer.

UserDefined: The license key is bound to the computer based on a user defined unique identifier.

EvaluationPerUser

Set this property to true to store evaluation information per user. The default value is true. If set to false, evaluation information is stored at the machine level. Note that you need to make sure the current user has the required privileges to store evaluation information at the machine level under HKEY_LOCAL_MACHINE\Software\Classes\CLSID\<GUID>.

Evaluation information consists of the installation date of your software as well as the last time your software ran.

C++: VARIANT_BOOL EvaluationPerUser

C#: bool EvaluationPerUser

ExpiryDate

Returns the expiry date of the evaluation key. You must call ValidateLicense prior to calling this function.

C++: DATE ExpiryDate

C#: System.DateTime ExpiryDate()

Return

Expiry date of the evaluation key.

FavorMachineLevelLicenseKey

This property affects the ReadKeys API.

If a license key is stored both at the machine level and user level, QLM will use the machine level key if this attribute is set to true. The default value is false.

For more details, read the help on how StoreKeys works.

Features

Returns an array of all the feature sets associated with the license key. Within a feature set (each element of the array), if several features are associated to a license key, the returned value is a bitwise OR of these features.

This function must be called after a call to ValidateLicense or ValidateLicenseEx.

C++: int *Features

C#:/STRONG> int [] Features

GetStatus

Returns the last status. See ELicenseStatus for possible values.

You must always call this function after calling ValidateLicense or ValidateLicenseEx to get the result of the validation.

C++: int GetStatus

C#: int GetStatus ()

Return

Last status

IsEvaluation

Returns whether the current license key is an evaluation key. You must call ValidateLicense prior to calling IsEvaluation.

C++: VARIANT_BOOL IsEvaluation

C#: bool IsEvaluation ()

Return

Boolean indicating if the license key is an evaluation key.

IsFeatureEnabled

Returns whether the specified feature is enabled in this license key. This function is now obsolete and has been superseded by IsFeatureEnabledEx.

You must call ValidateLicense prior to calling IsFeatureEnabled. **C++:** VARIANT_BOOL IsFeatureEnabled (int feature)

C#: bool IsFeatureEnabled (int feature)

Parameters

feature - id of feature to be checked.

Return

Boolean indicating if the featured is enabled.

IsFeatureEnabledEx

Returns whether the specified feature is enabled in this license key. You must call ValidateLicense prior to calling IsFeatureEnabled.

C++: VARIANT_BOOL IsFeatureEnabled (int featureSet, int feature)

C#: bool IsFeatureEnabledEx (int featureSet, int feature)

Parameters

featureSet - id of the feature set. QLM supports four feature sets (0 to 3).

feature - id of feature to be checked.

Return

Boolean indicating if the featured is enabled.

IsValid

Returns whether the current license key is a valid key. A valid license key is a key that was decoded properly and is either permanent or evaluation. You must call ValidateLicense prior to calling IsValid.

C++: VARIANT_BOOL IsValid

C#: bool IsValid ()

Return

Boolean indicating if the license key is a valid key.

LicenseType

Returns the license type of the key. See ELicenseType for possible values.

C++: ELicenseType LicenseType

C#: ELicenseType LicenseType ()

Return

License type

LimitTerminalServerInstances

This attribute limits the number of instances of your application when running on a Terminal Server.

When set to true, the QLM ValidateLicense API will fail if the number of running instances is greater than the one defined in the license key.

An instance is defined as an instance of your application with a unique user id and a unique session id.

For example, if a user initiates a remote desktop session to a system, and launches 5 instances of your application, the 5 instances will count as one because the same user has launched the 5 instances in the same session.

If the same user then initiates another remote desktop session and launches 3 instances of your application, the total number of instances will be 2.

This feature is only available with QLM Professional and QLM Enterprise.

You can control the number of allowed instances on a terminal server session by setting the Floating Seats property when creating an Activation Key.

NumberOfLicenses

Returns the number of multiple activations enabled for the license key.

C++: int NumberOfLicenses

C#: int NumberOfLicenses;

Return

Number Of LNumber Of Licenses

MajorVersion

Returns the major version associated to the license key. You must call ValidateLicense prior to calling this function.

C++: int MajorVersion

C#: int MajorVersion

Return

Major version of the product.

MinorVersion

Returns the minor version associated to the license key. You must call ValidateLicense prior to calling this function.

C++: int MinorVersion

C#: int MinorVersion

Return

Minor version of the product.

ParseResults

Parses the result of a web service call. All web service calls return an XML fragment describing the results of the call. This function parses the results and returns an ILicenseInfo interface describing the results.

C#: public bool ParseResults(string results, ref ILicenseInfo licenseInfo, out string message)

Parameters

results - value return from any web service call

licenseInfo - object containing the result of the parse.

message - error messages if an error occured while parsing the data

PrivateKey

QLM version 5 implements asymmetric encryption to encrypt the license key. Asymmetric enryption is safer because one key encrypts the license, the private key, and another key, the public key, decrypts that information. Therefore, you only need to include the public key in your source code.

This function sets the private key associated with your product. The private key should be set before you create a license, typically right after the call to DefineProduct. If you are creating a license key with a QLM engine version prior to version 5, you do not need to set the private key. It is highly recommended that you do not set the private key in your code.

The private key of your product can be found on the DefineProduct screen under the Keys tab in the QLM Console.

C++: _bstr_t privateKey

C#: string PrivateKey

ProductID

Returns the product ID associated to the license key. You must call ValidateLicense prior to calling this function.

C++: ibt ProductID

C#: int ProductID ()

Return

Product ID associated to the license key.

ProxyUser

Get or set the name of the user account to use when connecting via a proxy server.

You must also set the following properties: UseProxy, ProxyPassword and ProxyDomain.

ProxyPassword

Get or set the password of the user account to use when connecting via a proxy server.

You must also set the following properties: UseProxy, ProxyUser and ProxyDomain.

ProxyDomain

Get or set the domain of the user account to use when connecting via a proxy server.

You must also set the following properties: UseProxy, ProxyUser and ProxyPassword.

PublicKey

QLM version 5 implements asymmetric encryption to encrypt the license key. Asymmetric enryption is safer because one key encrypts the license, the private key, and another key, the public key, decrypts that information. Therefore, you only need to include the public key in your source code.

This function sets the public key associated with your product. The public key should be set before you validate a license, typically right after the call to DefineProduct. If you are validating a license key with a QLM engine version prior to version 5, you do not need to set the public key.

The public key of your product can be found on the DefineProduct screen under the Keys tab in the QLM Console.

C++: _bstr_t publicKey

C#: string PublicKey

ReadKeys

Reads the license keys stored on the end user system with the StoreKeys API. To store keys, use the StoreKeys API. To clear the stored keys, use the DeleteKeys API.

Use the FavorMachineLevelLicenseKey to control which key takes precedence if the keys are stored at the user and machine level.

C#: void ReadKeys (ref string activationKey, ref string computerKey)

Parameters

activationKey- the stored activation key.

computerKey- the stored computer bound key.

StoreKeys

Stores license keys on the end user system. To read keys, use the ReadKeys API. To clear the stored keys, use the DeleteKeys API.

QLM tries to stores the keys in two folders: one folder at the user level and another folder at the machine level.

If you are running XP, the folders are:

C:\Documents and Settings\<your account name>\Application Data\IsolatedStorage*
C:\Documents and Settings\All Users\Application Data\IsolatedStorage

On Vista, the folders are:

C:\ProgramData\IsolatedStorage
CC:\Users\ralph\AppData\Local\IsolatedStorage

Example on Vista:
C:\ProgramData\IsolatedStorage\1zy03lmk.jql\epxur3qn.na0\StrongName.gziza0ait44cgjtqq2fgdpi3yp0idvio\AssemFiles

Under these folders, a file whose name is the GUID associated to your product (GUID in Define Products page) is created and containts the license keys.

C#: void StoreKeys (string activationKey, string computerKey)

ParametersSTRONG>

activationKey- the activation key to store

computerKey- the computer bound key to store.

UseProxy

Get or set the flag to enable using a proxy server when connecting to the QLM web service.

When set to true, you must also set the following properties: ProxyUser, ProxyPassword and ProxyDomain

ValidateFile

Validates that the Quick License Manager DLL is authentic and was not tampered with. In order to prevent hackers from replacing the IsLicense50.dll with their own version, you can validate the authenticity of the DLL by calling the ValidateFile function. The ValidateFile function returns a fingerprint (long number) that is the result of a checksum of the DLL contents combined with your own key. Use the QlmFingerPrint.exe to generate this unique fingerprint and validate in your code that the runtime fingerprint matches the generated one.

If you are using QLM Professional, you do not need to call this function. Instead, set the validateIntegrity argument to true when constructing the QlmLicense object.

C++: long ValidateFile (BSTR LicenseDLL, BSTR Key);

C#: ulong ValidateFile(string LicenseDLL, BSTR Key)

Parameters

LicenseDLL: Full path to the License DLL. If this argument is NULL, the currently loaded License DLL is used.

Key: A unique key of your choice that is used to uniquely encrypt the fingerprint.

Return

FingerPrint- A long umber that uniquely identifies your license DLL.

ValidateLicense

Validates a license key. You must call DefineProduct prior to calling this function.

C++: _bstr_t ValidateLicense (BSTR LicenseKey);

C#: string ValidateLicense (string LicenseKey)

Parameters

LicenseKey: License Key to validate

Return

Error message if ValidateLicense fails to validate or if the license is an evaluation license.

ValidateLicenseEx

Validates a computer bound license key. You can call this function for any type of license key. If the license key is not computer bound, set the ComputerID to an empty string. You must call DefineProduct prior to calling this function.

C++: _bstr_t ValidateLicenseEx (BSTR LicenseKey, BSTR ComputerID);

C#: string ValidateLicenseEx (string LicenseKey, string ComputerID)

Parameters

LicenseKey: License Key to validate

ComputerID: A string identifying the computer. License Key to validate

Return

Error message if ValidateLicenseEx fails to validate or if the license is an evaluation license.

Version

Returns the version of the QLM engine used to create the key. You must call ValidateLicense prior to calling this function.

C++: _bstr_t Version

C#: string Version

Return

Version of QLM Engine used to create the license key.

WSCredentials

By default, QLM uses anonymous authentication to connect to the QLM web service.

To connect to the QLM web service via Windows Authentication, you must set the credentials of the user that will connect to the web service.

Note that to configure the QLM console to connect to your web service via Windows Authentication, you must set the Authentication method in the QLM Console / Sites property sheet.

Example:

```
QlmLicense license = new QlmLicense ();
WSCredentials ws = new WSCredentials ("user", "pwd", "domain");
license.WSCredentials = ws;
```

GetCPUIDs

Gets the ID of all CPUs found on the system.

C++: _bstr_t GetCPUIDs ();

C#: string GetCPUIDs ();

Parameters

None.

Return

Returns a semi colon separated list of CPU IDs.

GetFirstCPUID

Gets the ID of the first CPU found on the system. Note that some computer systems have several CPUs. This function returns the first CPU only. To get a list of all the CPUs on a system, use the GetCPUIDs method.

C++: _bstr_t GetFirstCPUID ();

C#: string GetFirstCPUID ();

Parameters

None.

Return

Returns the first ID of the first CPU found on the system.

GetFirstMACAddress

Gets the MAC address of the first network card found on the system. Note that some computer systems have several network cards. This function returns the MAC address of the first card only. To get a list of all MAC addresses on a system, use the GetMACAddresses method.

C++: _bstr_t GetFirstMACAddress ();

C#: string GetFirstMACAddress ();

Parameters

None.

Return

Returns the MAC address of the first network card found on the system.

GetMACAddresses

Gets the MAC addresses of all network cards found on the system.

C++: _bstr_t GetMACAddresses ();

C#: string GetMACAddresses ();

Parameters

None.

Return

Returns a semi colon separated list of MAC addresses.

GetVolumeSerialNumber

Gets the serial number of the specified volume.

C++: _bstr_t GetVolumeSerialNumber (_bstr_t volume);

C#: string GetVolumeSerialNumber (string volume);

Parameters

volume: volume driver letter such as C:.

Return

Returns the serial number of the specified volume.

RunningOnHyperV

Determines if the current process is running on a Microsoft Hyper-V virtual machine.

C++: BOOL RunningOnHyperV ();

C#: bool RunningOnHyperV ();

Parameters

None.

Return

Returns true if the current process is running on a Microsoft Hyper-V virtual machine.

RunningOnVirtualBox

Determines if the current process is running on a Virtual Boxv irtual machine.

C++: BOOL RunningOnVirtualBox ();

C#: bool RunningOnVirtualBox ();

Parameters

None.

Return

Returns true if the current process is running on a Virtual Box virtual machine.

RunningOnVM

Determines if the current process is running on a virtual machine such as Microsoft Hyper-V or VMWare.

C++: BOOL RunningOnVM ();

C#: bool RunningOnVM ();

Parameters

None.

Return

Returns true if the current process is running on a Virtual Machine such as Microsoft Hyper-V or VMWare.

RunningOnVMWare

Determines if the current process is running on a VMWare virtual machine.

C++: BOOL RunningOnVMWare ();

C#: bool RunningOnVMWare ();

Parameters

None.

Return

Returns true if the current process is running on a VMWare virtual machine.

HTTP Methods

The HTTP Methods are methods that can be invoked via a URL. These methods are typically invoked from your eCommerce provider during the purchase process.

Note that all other methods exposed by the QLM Web service cannot be called directly via SOAP. In order to communicate with the QLM Web service, you need to use the QLM .NET API methods that are exposed via the QLMLicenseLib.dll.

EnableMaintenancePlan

Enables the maintenance plan for a given activation key.

To invoke this method via a URL:

http://yourserver/yourvirtualdirectory/qlmservice.asmx/EnableMaintenancePlan?is_avkey=<*activationKey*>&*is_maintplan=1*&is_vendor=<*vendor*>&is_mainduration=<*duration*>

where

- is_vendor = One of the supported vendors
- is_avkey = Activation Key
- is_maintplan = Flag to determine if the maintenance plan should be extended. If this argument is specified , the maintenance plan is extended.
- is_maintdate = Date at which the new maintenance plan should expire
- is_maintduration = number of days by which to extend the maintenance plan as of today.
- is_user = username defined in the QlmProviders.xml (optional)
- is_pwd = password defined in the QlmProviders.xml (optional)

If neither is_maintdate nor is_mainduration are specified, the maintenance plan is extended based on the maintenancePlanPeriodInDays settings in the QLM web service config file (web.config). The default value of maintenancePlanPeriodInDays is 365 days.

If is_maintdate and is_maintduration are both specified, is_maintdate takes precedence.

The format of the date in *is_maintdate* is based on the *dateFormat* settings in the web service config file (web.config). The default format is: YYYY-MM-dd.

When invoking this method from an eCommerce provider, you can customize the url arguments for is_avkey and is_maintplan. Customization of these arguments requires subclassing of the eCommerce provider classes. For more details, contact us.

GetActivationKey

Creates an activation key over the internet.

Note that to call this function, you must update the web.config on the web server as follows:

<setting name="enableGetActivationKey" serializeAs="String">
<value>True</value>
</setting>

To invoke this method via a URL:
http://yourserver/yourvirtualdirectory/qlmservice.asmx/GetActivationKey?is_productid=<*productID*>
&is_majorversion=<*majorVersion*>&is_minorversion=<*minorVersion*>&is_vendor=*digibuy*&is_features=
<*features*>

where

- is_vendor = One of the supported vendors
- is_productid = your product id as defined in QLM
- is_majorversion = your product's major version as defined in QLM
- is minorversion = your product's minor version as defined in QLM
- is_qlmversion = 5.0.00 or earlier versions
- is_features = semi comma separated list of feature sets and ther corresponding values. Example: is_features=0:1;1:2;2:3;3:6
- is_usemultipleactivationskey = when set to true or not set, if multiple licenses are ordered in the same request, the system returns one activation key for all licenses. When set to false, the system returns one activation key for each ordered license.
- is_user = username as defined in the eCommerce Providers section in QLM (Manage Keys / Tools / eCommerce Providers)
- is_pwd = password as defined in the eCommerce Providers section in QLM (Manage Keys / Tools / eCommerce Providers)

GetActivationKeyWithExpiryDate

Creates an activation key with an expiry date over the internet.

Note that to call this function, you must update the web.config on the web server as follows:

<setting name="enableGetActivationKey" serializeAs="String">
<value>True</value>
</setting>

To invoke this method via a URL:
http://yourserver/yourvirtualdirectory/qlmservice.asmx/GetActivationKeyWithExpiryDate?is_productid=<*productID*>&is_majorversion=<*majorVersion*>&is_minorversion=<*minorVersion*>&is_vendor=*digibuy*&is_features=<*features*>&is_expduration=<*duration*>&is_expdate=<*date*>

where

- is_vendor = One of the supported vendors
- is_productid = your product id as defined in QLM
- is_majorversion = your product's major version as defined in QLM
- is minorversion = your product's minor version as defined in QLM
- is_qlmversion = 5.0.00 or earlier versions
- is_features = semi comma separated list of feature sets and ther corresponding values. Example: is_features=0:3;1:0;2:7;3:25
- is_usemultipleactivationskey = when set to true or not set, if multiple licenses are ordered in the same request, the system returns one activation key for all licenses. When set to false, the system returns one activation key for each ordered license.
- is_expduration = duration of evaluation version
- is_expdate = date at which the license will expire
- is_user = username as defined in the eCommerce Providers section in QLM (Manage Keys / Tools / eCommerce Providers)
- is_pwd = password as defined in the eCommerce Providers section in QLM (Manage Keys / Tools / eCommerce Providers)

The format of the date in *is_expdate* is based on the *dateFormat* settings in the web service config file (web.config). The default format is: YYYY-MM-dd

ReleaseLicenseHttp

Releases a license key over Http. This API releases a license key so that it can be activated on another computer.

To invoke this method via a URL:

http://yourserver/yourvirtualdirectory/qlmservice.asmx/ReleaseLicenseHttp?is_avkey=<*activationKey*>&is_pcid=<*computer ID*>&is_vendor=<*eCommerce provider*>

where

- is_avkey= Activation key to validate. If the key has never been activated, the key will be activated and a computer key will be returned.
- is_pcid = If the license has never been validated, you need to specify a computer identifier so that the returned computer key can be bound to this specific computer. A computer ID can be the name of the computer or any other unique identifier of your choice.
- is_vendor = name of the eCommerce provider

RenewSubscriptionHttp

Connects to the web service and upgrades a license. You can upgrade the following data associated to a license:

- Features associated to a license
- Expiry date of the license
- Duration of the license
- Major and Minor version of the product
- The version of the QLM Engine used to generate the license key

When a license is upgraded, a new license key is generated and replaces the existing one. The old license is copied to the released licenses table.

To use a proxy server, you must set the UseProxyServer, ProxyUser, ProxyDomain and ProxyPassword properties prior to calling this function.

To invoke this method via a URL:
http://yourserver/yourvirtualdirectory/qlmservice.asmx/RenewSubscriptionHttp?is_avkey=<*activationKey*>&is_vendor=<*xyz*>&is_expdate=<*date*>&is_user=<*user*>&is_pwd=<*pwd*>

where

- *is_vendor = One of the supported vendors*
- *is_expdate = date at which the license will expire*
- *is_user = username as defined in the eCommerce Providers section in QLM (Manage Keys / Tools / eCommerce Providers) - This argument is optional*
- *is_pwd = password as defined in the eCommerce Providers section in QLM (Manage Keys / Tools / eCommerce Providers) - This argument is optional*

SubscribeToMailListHttp

Subscribes or unsubscribes a user from the mail list.

To invoke this method via a URL:

http://server/qlm/qlmservice.asmx/SubscribeToMailListHttp?is_email=user@cie.com&is_include=1

where

- is_email = email of customer
- is_include = 1 to subscribe, 0 to unsubscribe

UpgradeLicense

Upgrades a license by issuing a new license key and replacing the old one. You can upgrade the following data associated to a license:

- Features associated to a license
- Expiry date of the license
- Duration of the license
- Major and Minor version of the product
- The version of the QLM Engine used to generate the license key

Note that to call this function, you must update the web.config on the web server as follows:

```
<setting name="enableUpgradeLicense" serializeAs="String">
<value>True</value>
</setting>
```

To invoke this method via a URL:
http://yourserver/yourvirtualdirectory/qlmservice.asmx/UpgradeLicense?is_productid=<*productID*>&is_majorversion=<*majorVersion*>&is_minorversion=<*minorVersion*>&is_vendor=*digibuy*&is_features=<*features*>&*is_expduration*=<*duration*>&*is_expdate*=<*date*>

where

- *is_vendor = One of the supported vendors*
- *is_productid = your product id as defined in QLM*
- *is_majorversion = your product's major version as defined in QLM*
- *is minorversion = your product's minor version as defined in QLM*
- *is_qlmversion = 5.0.00 or earlier versions*
- *is_features = semi comma separated list of feature sets and ther corresponding values. Example: is_features=0:3;1:0;2:7;3:25*
- *is_expduration = duration of evaluation version*
- *is_expdate = date at which the license will expire*
- *is_user = username as defined in the eCommerce Providers section in QLM (Manage Keys / Tools / eCommerce Providers)*
- *is_pwd = password as defined in the eCommerce Providers section in QLM (Manage Keys / Tools / eCommerce Providers)*

ValidateLicenseHttp

Validates a license key over Http. This API validates and activates a license.

The first time you call ValidateLicenseHttp, you provide the activation key (is_avkey) and the computer identifier (is_pcid). The server does the following:

- Validates the license
- Verifies if the license has not been previously activated
- Activates the license
- Returns the computer key and the set of features that are enabled.

On subsequent calls to ValidateLicenseHttp, in addition to the previous arguments, you should set the computer key argument using the value returned from the first call. In this instance, the server does the following:

- Validates the license
- Verifies if the license has not been revoked
- Returns the status of the license and the set of features that are enabled.

To invoke this method via a URL:

http://yourserver/yourvirtualdirectory/qlmservice.asmx/ValidateLicenseHttp?is_avkey=<*activationKey*> &is_pckey=<*computer key*> &is_pcid=<*computer ID*> &is_computer_name=<*computer name*> &is_qlmversion=<*QLM Engine version*> &is_email=<*email of the customer associated to the key*> &is_userdata1=<*user data to associate to the key*> &is_affiliateid=<*affiliate to associate to the key*>

where

- is_avkey= Activation key to validate. If the key has never been activated, the key will be activated and a computer key will be returned.
- is_pckey = If the key has been previously activated, the ValidateLicense method returns a computer key. This computer key should then be used in subsequent calls to ValidateLicense in the is_pckey argument.
- is_pcid = If the license has never been validated, you need to specify a computer identifier so that the returned computer key can be bound to this specific computer. A computer ID can be the name of the computer or any other unique identifier of your choice.
- is computer_name= This argument is not required. It is used to easily identify a computer, in case the computer ID is a serial number such as the hard disk serial number.
- is_qlmversion = Version of the QLM engine. IT can be 5.0.00 or earlier versions.
- is_email = Email address of the customer associated to the license key. This argument is optional.
- is_userdata1 = User data to associate to the license key. This argument is optional.
- is_affiliateid = Affiliate to associate to the license key .This argument is optional.

.NET Control

Following is a list of all the properties that can be set on the QLM .NET Control.

Name	Description
QlmCloseButtonVisible	Show or hide to Close button
QlmComputerID	Set the computer ID to use when activating the license. This property should be typically set programmatically at runtime.
QlmEncryptionKey	Set the encryption key. This property is only required when using QLM engine version 4.0 and earlier.
QlmEvaluationHaveKeyRadioButtonText	Only applies if the QlmEvaluationVisible property is set to true. Set the text in the radio button when the user has a license key.
QlmEvaluationLicenseKey	Only applies if the QlmEvaluationVisible property is set to true. Set the evaluation key to use when the user selects to evaluate the software and does not have a license key.
QlmEvaluationTrialChecked	Only applies if the QlmEvaluationVisible property is set to true. Checks the evaluation option by default.
QlmEvaluationTrialHelpText	Only applies if the QlmEvaluationVisible property is set to true. Set the text to display under the evaluation radio button.
QlmEvaluationTrialRadioButtonText	Only applies if the QlmEvaluationVisible property is set to true. Set the text to display on the evaluation radio button.
QlmEvaluationVisible	Enables the evaluation option. The evaluation option displays two radio buttons. One radio button allows the user to enter a license key and activate the license while the other radio button allows the user to evaluate the software by using an embedded evaluation license key.
QlmFormBackColor	Set the starting Background Color of the form to produce a gradient effect.
QlmFormBackColor2	Set the ending Background Color of the form to produce a gradient effect.
QlmGUID	Set the GUID associated to your product. The GUID can be found on the Define Product page in the QLM Console.
QlmHeaderBackColor	Set the Background Color of the header pane.
QlmLicenseStatus	Get the status of the license after it has been validated. This is a read-only property.
QlmLicenseType	Set the license type. The license type can be: ComputerName, UserDefined or Generic.
QlmLogoFont	Set the font to use in the logo text.
QlmLogoImage	Set the image to use for the logo.
QlmLogoText	Set the text to use for the logo.
QlmMajorVersion	Set the Major Version associated to your product. The Major Version can be found on the Define Products page in the QLM Console.

QlmMinorVersion	Set the Minor Version associated to your product. The Minor Version can be found on the Define Products page in the QLM Console.
QlmProductID	Set the Product ID Version associated to your product. The Product ID can be found on the Define Products page in the QLM Console.
QlmProductName	Set the Product Name associated to your product.
QlmProxyButtonVisible	Show or hide the proxy settings button
QlmPublicKey	Set the Public Key associated to your product. The Public Key Version can be found on the Define Products page (Keys tab) in the QLM Console.
QlmStoreKeysLocation	By default, QLM stores the license keys in a hidden file on the end user system. You can also select to store the license keys in the registry by setting this property.
QlmValidateCertificate	The QLM DLLs are digitally signed by a trusted certificate authority. In order to ensure that hackers do not replace the QLM DLLs by dummy ones, QLM can validate that the DLLs are properly signed.
QlmWebCommunicationEncryptionKey	Set the communicationEncryptionKey to use when connecting to the QLM web service. The communicationEncryptionKey must match the one defined in the web.confi file on the web server.
QlmWebServiceUrl	Set the URL to the QLM web service. The value of this url is typically: http://yourdomain.com/qlm/qlmservice.asmx

Floating Licenses

Quick License Manager Enterprise offers all the features of QLM Professional and in addition enables you to implement floating licenses.

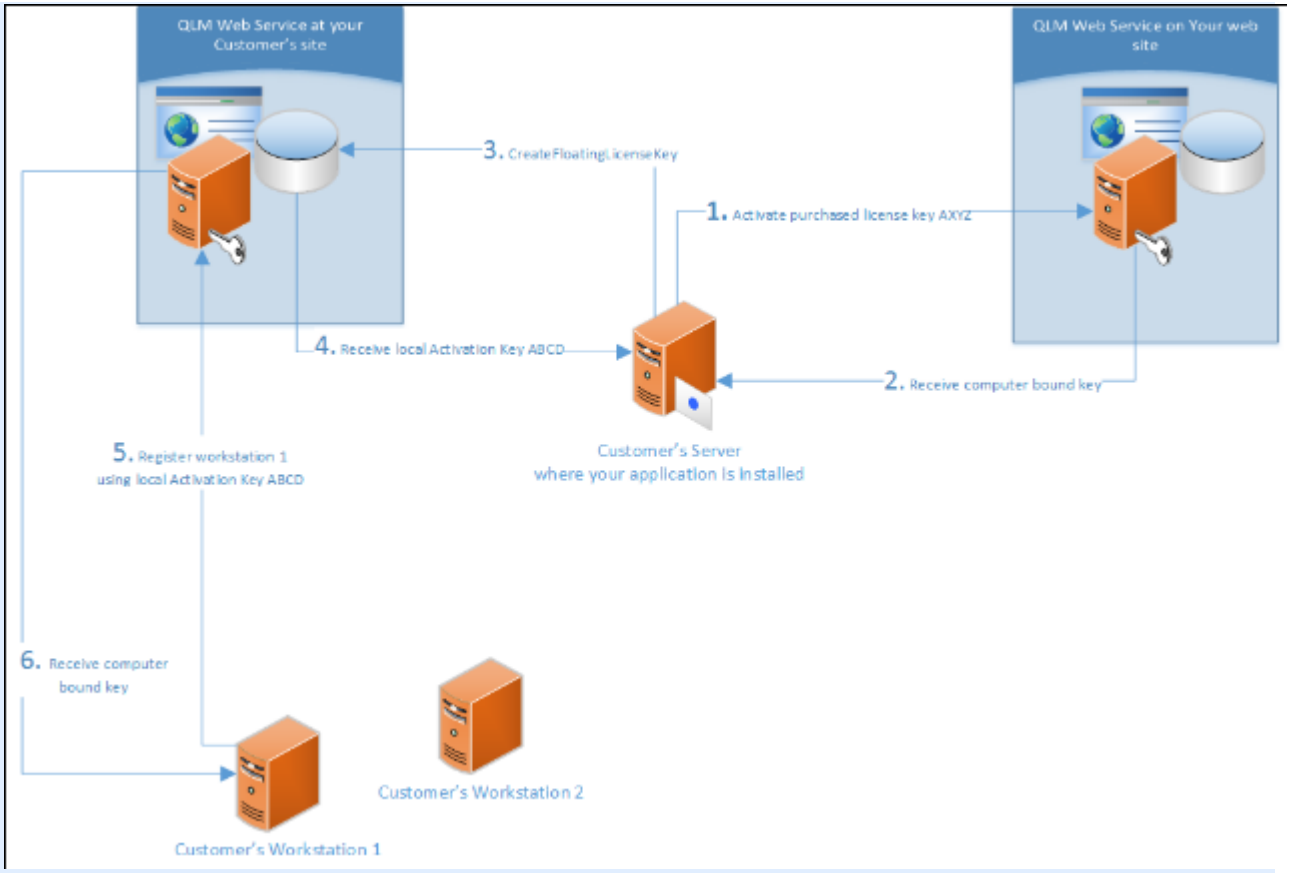
To implement floating licenses, you need to distribute the QLM web service with your application. You can either include the qlmwebsvcsetup.exe with your application and invoke it directly at your customer site or alternatively, you can repackage all the files in the DeployToServer\QLMWebService folder and install these files as described in **Web Service** section above.

Below are the steps required to add floating license support to your application:

- Upon purchase, issue an Activation Key to your customer. When creating the activation key, set the FloatingSeats property to the number of seats purchased. If you are issuing the activation key from your eCommerce provider via a URL, set the is_floating argument to true. If you are creating the activation key via the QLM Console, set the Floating Seats field to the number of purchased seats.
- Your customer installs your product and the QLM web service.
- As part of the QLM web service installation, you should install a QLM database at the customer site. The QLM database at the customer site should include in the Products table details about your product. It is critical that the Private/Public key pair in your own database be different than the one in the customer's database. Similarly, the product ID and version numbers of your actual product do not need to match the ones in the customer's database.
- Your customer enters the Activation Key in your product and clicks on Activate.
- Your application connects to the QLM web service that you are hosting (over the internet) and issues a computer bound key to the customer.
- You then store this key and call CreateFloatingLicenseKey function as follows:
    - Call DefineProduct
    - Call ValidateLicense
    - Call CreateFloatingLicenseKey
    - Make sure you call this function only once.
    - *The QLM web service used in this call and all subsequent calls should be the local QLM web service installed at the customer site and not the QLM web service you are hosting.*
- The CreateFloatingLicenseKey returns a license key. This key should be used by all nodes to access the system.
- Every time a node starts, you should call the ActivateLicense function by providing the key returned by CreateFloatingLicenseKey as the ActivationKey argument.
- Every time a node exits, you should call the ReleaseLicense function by providing the key returned by CreateFloatingLicenseKey as the ActivationKey argument.

A sample showing how to implemenet floating licenses is provided in the following folder:

%AppData%\My Documents\Quick License Manager\Samples\QLMEnterprise\DotNet\C#\VS2005\QlmEntSample


QLM Web Service at your Customer's site
QLM Web Service on Your web site
3. CreateFloatingLicenseKey
1. Activate purchased license key AXYZ
4. Receive local Activation Key ABCD
2. Receive computer bound key
5. Register workstation 1 using local Activation Key ABCD
Customer's Server where your application is installed
6. Receive computer bound key
Customer's Workstation 2
Customer's Workstation 1

CreateFloatingLicenseKey

Creates a floating license key based on the currently validated key. To use this function, call DefineProduct, then ValidateLicense. If ValidateLicense is successfull, call CreateFloatingLicenseKey. This function should only be called once. If you call it multiple times, you will end up creating multiple floating licenses. Store the license key returned by this call. All clients of this floating license, should use this key to activate their nodes. To use a proxy server, you must set the UseProxyServer, ProxyUser, ProxyDomain and ProxyPassword properties prior to calling this function.

C#: CreateFloatingLicenseKey(string webServiceUrl, string email, string features, int numSeats, string qlmVersion, string userData1, DateTime expiryDate, int expiryDuration, out string results)

Parameters

webServiceUrl - URL to the QLM Web service.

email - customer's email.

features - semi comma separated list of feature sets and ther corresponding values. Example: is_features=0:1;1:2;2:3;3:6

numSeats - Number of seats

qlmVersion - Version of the QLM Engine to use. Current version is: 5.0.00

userData1 - User data to associate with the license key

expiryDate - Expiry date of the license

expiryDuration - Expiry duration of the license

response - XML fragment containing the result of the call. The Xml fragment schema is as follows:

```
<?xml version='1.0' encoding='UTF-8'?>
<QuickLicenseManager>
<result>ActivationKey A162DCF05C30D371A2D0E0461040A0 has been created.</result>
</QuickLicenseManager>
```

In the event of an error, the XML fragments returns:

```
<?xml version='1.0' encoding='UTF-8'?>
<QuickLicenseManager>
<error>Details about the error</error>
</QuickLicenseManager>
```